A finite element continuum model for consolidation due to pumpage.

Kang, Jae-Sun

CIVIL AND COMPUTATIONAL ENGINEERING CENTRE
| SCHOOL OF ENGINEERING
UNIVERSITY OF WALES SWANSEA

A FINITE ELEMENT CONTINUUM MODEL FOR CONSOLIDATION DUE TO PUMPAGE

JAE-SUN KANG

B.Sc. M.Sc. (Pusan National University)

THESIS SUBMITTED TO THE UNIVERSITY OF WALES IN CANDIDATURE FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

October 2005

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A FINITE ELEMENT CONTINUUM MODEL FOR CONSOLIDATION DUE TO PUMPAGE

DRAFT OF PH.D. THESIS

JAE-SUN KANG

OCTOBER 2005

CIVIL AND COMPUTATIONAL CENTRE

SCHOOL OF ENGINEERING

UNIVERSITY OF WALES SWANSEA

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date ..28th December, 2005.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date ..28th December, 2005.....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date ..28th December, 2005.....

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF FLOW CHARTS

SUMMARY

Modelling the mechanical behaviour of engineering phenomena has occupied the attention of researchers since Karl Terzaghi's pioneering work on settlement due to consolidation in 1925. Soil improvement methods using vertical drains in combination with pre-loading were used for soil stabilization at the San Francisco-Oakland Bay Bridge in 1926. Over the next decades, other drain methods, including prefabricated and pack drains, were developed and used for practical purposes. In the mid-1990s, vacuum-induced consolidation became a reliable technology, thanks to a rethinking of theoretical principles that led to the Menard vacuum system. The Menard system was used successfully, for the first time in South Korea, for construction of the Kimhae sewage treatment plant.

The drain method is based on Terzaghi's one-dimensional consolidation theory and Barron's vertical drain. Even when using finite element simulation, numerical solution for consolidation problems involves averaging material properties such as elastic modulus, Poisson's ratio, and coefficients of permeability. But the results are too unreliable to use for practical purposes. Classical mechanics alone do not provide sufficient information on the global motion equation of fluid in a porous media.

A new approach is presented in this paper to the problem of continuum modelling of vacuum-induced consolidation due to pumpage. This finite element continuum model uses the interior boundary condition instead of installed vacuum tube as a line, with changing pore pressures at the installed material being treated as boundary conditions of the interior part. An innovative linear equation solution method for separate fixed boundary conditions is presented.

The efficacy of this model for field construction is shown by comparing results with the results obtained from field measurements at the Jangyoo sewage plant. Because the properties of material are not exact in their natural states, the results of the calculated finite modelling are similar but not a mirror image of field measurements. Whereas conventional one-dimensional calculation uses only one point, the finite element continuum model shows displacements and pore pressures for a whole section. Once the exact material properties have been determined, the model can be applied effectively to field analyses, predicting settlements due to pumpage and facilitating decision making about when pumpage should start and stop.

The finite continuum model for consolidation due to pumpage can be applied to other soil improvement methods, such as prefabricated and pack vertical drain, with some modification.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor, Professor Roland W. Lewis, for his guidance, advice and encouragement throughout my study in Swansea. I would also like to thank him for looking after my family during our year in Swansea. I am grateful to invited Professor Mary Lasovich at Kyung Nam university for her kindness and help with English writing.

I would like to express my appreciation to all the staff members at my department for their technical support.

Among all the people I have met in the last nine years at Swansea University I want to especially thank Professor G. N. Pande and D. J. R. Owen. I also want to thank Dr. D. V. Tran, H. R. Ghafouri, R. S. Ranshing and many other colleagues in my department for being good friends. I would like to express my appreciation to all the Korean seniors, Dr. Hyu-Soung Shin, Dr. Sang-Jin Lee, and Mr. Kwan-Ho Lee. I would also like to thank other Korean families in Swansea for their kindness and help.

Finally, I am deeply indebted to my mother-in-law who has supported me for a long time. Especially, I am obliged to my wife Min-Ok Kim and my son Tae-Han Kang for their unconditional patience and deep trust in me during my years of study. Without their understanding, I would have never been able to finish this work.

Notation

Romanic symbols and their meaning

a_v = the coefficient of compressiblility.

$a\left(\varepsilon_v^p\right)$ is the semi-diameter of the ellipse in the p=direction

a and b are a set of arbitrary functions since $\overline{A}$ and $\overline{B}$ are identically satisfied throughout their respective domains

B_x, B_y, B_z denote the components of this force per unit volume of the element

B_g = gas formation volume factor

B_0 = oil formation volume factor

B_t = total formation volume factor

B_w = water formation volume factor

c_v = coefficient of vertical consolidation

c_h = coefficient of horizontal consolidation

C_s in the specific moisture content is defined in terms of the pressure instead of the pressure head $\psi = p_w / \gamma_w$

c = apparent cohesion

c_m is the uniaxial compaction coefficient

c_r is the rock matrix compressibility

c_k is coefficient of permeability reduction

C_f = effective rock compressibility

C_w = water compressibility

c_b = the bulk compressibility

d = length of longest drainage path

d = drain diameter

d is the initial depth

D^e is the tangential elastic stiffness matrix

e = void ratio

e_1 is the initial void ratio

e_2 is the final void ratio

E = Young's modulus

F = cumulative fluid injection

$G = \dfrac{E}{2(1+\nu)}$ is shear modulus

g is the gravity

$\overline{g}$ is an arbitrary function of time

G = initial free gas in place

G_p = cumulative gas production

h is the head above some arbitrary datum

$I_1 = \sigma_x + \sigma_y + \sigma_z$ is the first stress invariant

k = permeability

K_w is the bulk modulus of water

$k_{r\pi}$ is a function of the saturation of the $\pi - phase$

K_w is the bulk modulus of water

m_v = coefficient of compressibility = $\dfrac{a_v}{1+e}$

n = any integer which lies within the limits $-\pi/6 \le \theta_0 \le \pi/6$

n_x is the direction cosine between the outward normal and the $x - direction$

N = initial oil in place

N_p = cumulative oil production

n is the unit normal vector

$n(\theta)$ is the ratio of the diameters in the q and p=directions, defined by the Mohr-Coulomb equation for c=0

p = pore pressure

p is the fluid pressure

$\overline{p}$ is the average effective pressure

p_π denotes the pressure

$\overline{p} = p_w$

p_0 is the pressure in the oil phase

$\overline{p}$ = average reservoir pressure, average pressure at initial oil-water contact

q is the outflow rate per unit area of the boundary surface

Q is the potential surface

R_{so} = gas-oil solution ratio

S_w = average interstitial water saturation

t = time space

t = time

T_h = time factor

Δt_k is the length of the k^{th} time step

T refers to stress with atmospheric pressure

t is stress without atmospheric pressure

u_i = initial excess pore water pressure

u = excess pore pressure

U = degree of total consolidation

U_h = degree of horizontal consolidation

U_v = degree of vertical consolidation

$\overline{V_\pi}$ is the volume of the dissolved non-wetting $\pi - fluid$ at pressure P_π

V_{dgSTC} is the volume that the gas dissolved in unit mass of the $\pi - phase$ would occupy at STC, and $V_{\pi STC}$ is the volume of the $\pi - phase$ at STC

W is a set of weighting functions in terms of the local coordinates ξ, η and ζ

W_e = water influx from the adjacent aquifer and from leaky aquitards

W_p = cumulative water production

z = vertical coordinate

Greek symbols and their meaning

α_i = constant

$\alpha_p\left(\varepsilon_v^p\right)$ is the p-coordinate of the centre of the elliptical surface

β is the ratio of rock matrix and rock bulk compressibility

δ is the average settlement of each element

ς_π is the value of ς within the $\pi - phase$

θ_0 is a part of the general solution to equation (4.4)

μ is the dynamic viscosity of the fluid

ρ is the density

σ = increment of pore water pressure

$\boldsymbol{\sigma}$ are total stresses as $\left\{\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\right\}^T$

σ' are effective stresses as $\left\{\sigma'_x, \sigma'_y, \sigma'_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\right\}^T$

σ_1 and σ_3 are the major and minor principal stresses at failure

σ'_i is the effective stress before vacuum

τ_π denotes the shear stress vector

ϕ = angle of internal friction

ψ is the dilatancy angle

γ is the specific weight of solid

γ_w = unit weight of water

x = vertical space coordinate

$\chi = \lambda - k/1 - e_0$ is an empirical constant

ν is Poisson's ratio

$\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{\partial^2}{\partial z^2}$ the Laplace differential operator

Indices. Subscript. Superscript. Abbreviation

i = subscript: denotes initial conditions

Chapter 1

Introduction

1.1 Motivation

Modelling the mechanical behaviour of engineering phenomena has occupied the attention of researchers since Karl Terzaghi's pioneering work on settlement due to consolidation in 1925. Since then, other drain methods have been developed and used for practical purposes. Attention has now shifted to the problem of modelling consolidation due to pumpage.

Numerical modelling such as the finite element (FE) method plays an important role as a tool for analysing variety problems in engineering. One of the most crucial aspects of FE analysis is the consolidation problem.

Even when using finite element simulation, numerical solution for consolidation problems involve averaging material properties such as elastic modulus, Poisson's ratio, and coefficients of permeability. But the results are too unreliable to use practical purposes. Classical mechanics alone do not provide sufficient information on global motion equation of fluid in porous media.

A new approach is presented in this thesis to the problem of continuum modelling of vacuum-induced consolidation due to pumpage. This finite element continuum model uses the interior boundary condition instead of installed vacuum tube as a line, with changing pore pressures at the installed material being treated as boundary conditions of the interior part. An innovative linear equation solution method for separate fixed boundary conditions is presented.

1.2 objectives

The objectives of this thesis are:

- To develop a solving method of linear equation system with fixed

boundary condition.

- To treat vacuum tubes with fixed boundary condition.

- To verify the developed numerical method by comparing the results obtained the field measurements of a practical ploblem constructed using the Menard vacuum method.

Each chapter in this thesis stands on its own objectives.

1.3 Layout of theses

Chapter 2 summarize of the basic theories of one- and three-dimensional consolidation and introduce consolidation by pumpage.

Chapter 3 presents the one dimensional governing equations of Terzaghi's pioneering work, the three dimensional one developed by Biot, and Barron's vertical drain consolidation theory. The governing equations for multiphase flow in porous media are also described.

Chapter 4 addresses constitutive equations and the variable permeability.

Chapter 5 introduces numerical solution of the continuum theory for the coupled problem, which is of concern in consolidation theory.

Chapter 6 moves into the heart of this thesis, presenting a new modelling and simulation approach in the context of procedures and principles of a continuum model for consolidation due to pumpage. The theory of vacuum-induced consolidation, finite element modelling, the idea of interior boundary condition, and a linear equation solution for modelling consolidation due to pumpage are contained in this chapter. A flow chart of programs used in finite element application is also presented.

Chapter 7 provides verification of the developed program through comparision with the PLAXIS program, showing the results of y-directional displacements for a simple consolidation problem.

Chapter 8 describes a practical application of the Menard vacuum system at a sewage treatment plant in Kimhae, South Korea.

Chapter 9 presents a detailed modelling of the Jangyoo project, showing the accuracy of the proposed finite element continuum model for determining settlements, pore pressures, and horizontal displacement. Numerical results obtained with the solution method are shown to compare favourably with field measurements.

Chapter 10 presents conclusions and ideas for future research.

Chapter 2

A survey of consolidation analysis

Analysis methods for consolidation first took analytical form with Karl Terzagi's one-dimensional consolidation theory, and were then extended by the three-dimensional theory developed by M. A. Biot in 1941. Numerical methods for solving consolidation problems have been developed by many authors. The finite difference method was the first numerical technique to transform the differential operator to the finite difference, but some difficulties arose when this method was applied to irregular domain and many different materials. The finite element solution method proved more adaptable than the finite difference technique because this method treats irregular domains and many different material properties. Nowadays, the finite element method of numerical analysis is very popular.

2.1 Theoretical aspects.

The subsidence of weak ground by consolidation involves a process by which pore water is squeezed from the clay soil. The theory of consolidation was developed to analyze the conditions for the flow of pore water from soil aggregate. The basic assumptions of Terzaghi's one-dimensional consolidation theory are as follows:[31,87,88]

(1) The voids of soil are completely filled with water.
(2) Both water and the solid constituents of soil are completely incompressible.
(3) Darcy's law is strictly valid.
(4) The coefficient of permeability k is a constant.
(5) Soil is homogeneous.
(6) The action of infinitesimal masses is no different from that of larger, representative masses.
(7) Soil is one-dimensional compressive.

(8) The pore water flow is one dimensional.
(9) The relation of pore pressure versus void ratio is linear.

These assumptions are idealized in the one-dimensional partial differential equation used for consolidation problems, but they do not fully account for natural phenomena.

More than 60 years ago, Biot advanced our knowledge by developing a general, three-dimensional consolidation theory that is more closely linked to natural conditions than Terzaghi's work. Key assumptions made by Biot in constructing his theory are as follows:[8]

(1) Material is isotropic.
(2) Reversibility occurs in stress-strain relations under finial equilibrium conditions.
(3) Linearity occurs in stress-strain relations
(4) Theory of small strain is valid.
(5) The water that may be contained in the pore is incompressible.
(6) The water may contain air bubbles.
(7) The water flows through the porous skeleton according to Darcy's law.

The theories developed by Terzaghi and Biot laid the groundwork for analytical problem solving, but were limited to simple equations. Computer technology has now made it possible to undertake complex numerical solution methods.

In the late 1940s, R. A. Barron developed a theory of consolidation for drain wells that led to improvement in weak foundation soils and greater understanding of vertical drain, such as sand, prefabricated, and pack drain methods, as well as the pumpage of pore water from weak soils. Analysis of vertical drain methods is based on Barron's theory[4] and if soils have many layers, soil properties must be changed to average one. The pumpage method is somewhat different from the other drain methods, in that load of embankment plus atmospheric pressure are used to calculate settlement.

2.2 Numerical analysis techniques.[3,31]

Numerical analysis can be defined as the solution of a physical problem by arithmetic means. A physical problem is usually modelled on a symbolic mathematical relationship. Numerical analysis is the means by which arithmetic operations are used to provide numerical results for engineering use. For consolidation, the symbolic form is a partial differential equation governing the behaviour of the excess pore pressure u as a function of space and time. For a given set of physical parameters and boundary and initial conditions, the numerical analysis provides numerical values for the excess pore pressure.

Three types of numerical analysis can be applied to consolidation problems. The first concerns the evaluation of analytical solutions by numerical means. This involves the evaluation of the roots of a characteristic equation and the summation of a series over the characteristic values. The second numerical technique is a finite difference procedure, in which the governing differential equation is represented by a set of difference equations which are then solved algebraically. The third type of analysis is by finite elements, in which the consolidating mass is represented by compatible elements and problem solving requires the application of a variational principle. Numerically, the finite element procedure involves the solution of a set of simultaneous linear algebraic equations. Another useful approach to consolidation problem is the boundary element method, in which boundary parts of soils are treated discretely to reduce dimensionality. This method is useful only when the soil stratum is homogeneous. The most useful of the three methods is the finite element method because of its utility and applicability to many kinds of problems. All of these numerical methods require computation by machine procedures.

2.3. Consolidation caused by pumpage effects.[16,47]

Improvement of settlements due to consolidation has been an important issue in many parts of the world, including South Korea, China, and other Southeast Asian countries. Popular methods of improvement for weak

soils include vertical drain methods (sand, prefabricated, and pack drain) and the more rapid pumpage method developed by the Menard Co. of France. Pumpage is different from vertical drain methods because it involves forced drainage of pore water in the weak foundation soils by injected tubing and pumping.

Initial calculation of settlements of weak ground using Terzagi's one-dimensional consolidation theory have been improved by adding vacuum pressure as atmospheric pressure to embankment loads. The effect of pumpage has also seen to have an important role in improving weak grounds and promoting the settlement of clayey ground. The prediction method of ground settlement due to pumpage uses both field measurements and a method developed by Asaoka to determine when to stop pumpage.[2] This method is very useful to get rapid settlement of 2 to 5 times faster than the other methods.

2.4 Concluding remarks.

This chapter has reviewed the theoretical development of consolidation problems for weak soils. Numerical methods for solving settlement of weak ground were described as well as the fundamental methodology to calculate settlement of weak ground.

Chapter 3

Theories and governing equations of consolidation

Governing equations of consolidation are presented in this chapter, illustrating how the calculations for Terzaghi's one-dimensional consolidation theory have been adapted to account for displacement and compatibility relations that impact on pore water flow. Next, a general formulation of multiphase flow in a deforming porous medium is shown at both macroscopic and microscopic levels.

3.1 Terzaghi's one-dimensional consolidation theory.

The theory for time rate of one-dimensional consolidation was first proposed in 1925 by Terzaghi who developed mathematical equations from a set of assumptions listed in the previous chapter (2.1). The governing equation for Terzaghi's theory is as follows (3.1):

$$\frac{k}{\gamma_w}\frac{\partial^2 u}{\partial z^2} = \frac{a_v}{1+e}\frac{\partial u}{\partial t} = m_v \frac{\partial u}{\partial t} \tag{3.1}$$

where

k = permeability.

γ_w = unit weight of water.

$u = h\gamma_w$.

h = height of pore water.

e = void ratio.

z = vertical coordinate.

t = time space.

a_v = the coefficient of compressibility.

m_v = coefficient of compressibility = $\frac{a_v}{1+e}$ (3.2)

The total stress increment is assumed to be applied instantaneously. At zero time, therefore, the increment will be carried entirely by the pore water, making the initial value of excess pore water pressure (u_i) equal to stress increment ($\Delta\sigma$). The initial condition is

$$u_e = u_i \text{ for } 0 \le z \le 2d \text{ when } t = 0 \quad (3.3)$$

The upper and lower boundaries of the clay layer are assumed to be free-draining, the permeability of the soil adjacent to each boundary being very high compared to that of the clay. Thus, the boundary conditions at any time after the application of $\Delta\sigma$ are

$$u_e = 0 \text{ for } z = 0 \text{ and } z = 2d \text{ when } t > 0 \quad (3.4)$$

The solution of the equation (1) for the excess pore water pressure at depth z after time t is

$$u_c = \sum_{n=1}^{n=\infty} \left(\frac{1}{d} \int_0^{2d} u_i \sin\frac{n\pi z}{2d}\, dz\right)\left(\sin\frac{n\pi z}{2d}\right)\exp\left(-\frac{n^2\pi^2 c_v t}{4d^2}\right) \quad (3.5)$$

Where

d = length of longest drainage path,

u_i = initial excess pore water pressure, in general a function z.

c_v = being defined as coefficient of consolidation. (m^2 / year)

$$= \frac{k}{m_v \gamma_w}$$

For the particular case in which u_i is constant throughout the clay layer,

the calculation is

$$u_e = \sum_{n=1}^{n=\infty} \frac{2u_i}{n\pi}(1-\cos n\pi)(\sin\frac{n\pi z}{2d})\exp(-\frac{n^2\pi^2 c_v t}{4d^2}) \tag{3.6}$$

When n is even, $(1-\cos n\pi)=0$, and n is odd, $(1-\cos n\pi)=2$. Only odd values of n are therefore relevant, and it is convenient to make the following substitutions: $n=2m+1$ and $M=\frac{\pi}{2}(2m+1)$

It is also convenient to substitute a dimensionless number called time factor

$$T_v = \frac{c_v t}{d^2} \tag{3.7}$$

The equation (3.6) of the solution for the governing equation (3.1), then becomes

$$u_e = \sum_{m=0}^{m=\infty} \frac{2u_i}{M}(\sin\frac{Mz}{d})\exp(-M^2 T_v) \tag{3.8}$$

The degree of consolidation at depth z and time t can be obtained by substituting the value of u_e in the equation (3.8), $U_z = 1-\frac{u_e}{u_i}$, giving

$$U_z = 1-\sum_{m=0}^{m=\infty} \frac{2}{M}(\sin\frac{Mz}{d})\exp(-M^2 T_v) \tag{3.9}$$

In solving practical problems it is the average degree of consolidation (U) over the depth of the layer as a whole that is the consolidation settlement at time t being given by the product of U and the final settlement. The average degree of consolidation at time t for constant u_i is given by

$$U = 1 - \frac{\frac{1}{2d}\int_0^{2d} u_e dz}{u_i} = 1 - \sum_{m=0}^{m=\infty} \frac{2}{M^2} \exp(-M^2 T_v) \qquad (3.10)$$

The preceding equations are the exact solution of Terzaghi's governing equation (3.1).

3.2 Biot's three-dimensional consolidation theory. [8,9,10,11]

Biot developed his theory of three-dimensional consolidation under assumptions outlined in section 2.1. It found that when considering a small cubic infinitesimal element of the consolidating soil due to external stress, the stress field must satisfy the well-known equilibrium for solids.

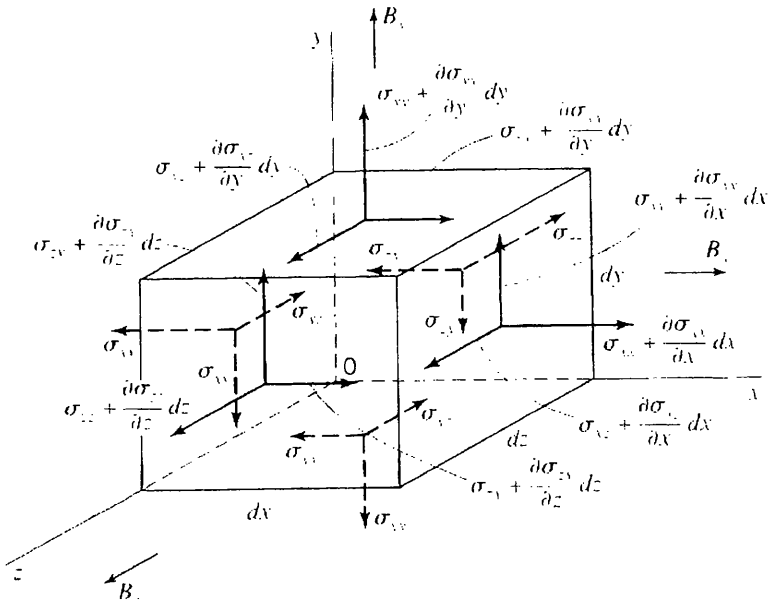


Figure 3.1 Stress components showing changes from face to face along with body force per unit volume including inertial forces.

$$\frac{\partial \sigma_x}{\partial x}+\frac{\partial \tau_{xz}}{\partial y}+\frac{\partial \tau_{xy}}{\partial z}+B_x=0,$$
$$\frac{\partial \tau_{yz}}{\partial x}+\frac{\partial \sigma_y}{\partial y}+\frac{\partial \tau_{yx}}{\partial z}+B_y=0, \qquad (3.11)$$
$$\frac{\partial \tau_{zy}}{\partial x}+\frac{\partial \tau_{zx}}{\partial y}+\frac{\partial \sigma_z}{\partial z}+B_z=0.$$

where B_x, B_y, B_z denote the components of this force per unit volume of the element.

These stresses are composed of two parts: one is caused by the hydrostatic pressure of the pore water; the other is caused by the average stresses in the soil skeleton. But in the solid mechanics, hydrostatic pressures are neglected.

Denoting as u,v,w the components of the displacement of the soil and assuming the strain to be small, the values of the strain components are

$$\varepsilon_x=\frac{\partial u}{\partial x},\gamma_x=\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}$$
$$\varepsilon_y=\frac{\partial v}{\partial y},\gamma_y=\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x} \qquad (3.12)$$
$$\varepsilon_z=\frac{\partial w}{\partial z},\gamma_z=\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}$$

The compatibility of the equation (3.11) is described as the compatibility equation (3.12). Combining these equations produces nine calculating equations, but the unknowns are 15, so another six equations are needed to solve the equilibrium equation. These six equations come from the constitutive behaviour from stress and strain relations.
The linear constitutive relation of solid is expressed mathematically for calculation purposes as

$$\boldsymbol{\sigma}=\mathbf{D}\boldsymbol{\varepsilon} \qquad (3.13)$$

where $\boldsymbol{\sigma}$ is $[\sigma_x \sigma_y \sigma_z \tau_{xy} \tau_{xz} \tau_{zy}]^T$, $\boldsymbol{\varepsilon}$ is $[\varepsilon_x \varepsilon_y \varepsilon_z \gamma_{xy} \gamma_{yz} \gamma_{zx}]^T$,

$$\mathbf{D} = \frac{E}{(1+\upsilon)(1-2\upsilon)} \begin{bmatrix} (1-\upsilon) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\upsilon) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\upsilon) & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu) \end{bmatrix} \tag{3.14}$$

where E=Young's modulus and ν= Poisson's ratio.

If the displacements (u,v,w) and the strains $(\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{xy}, \gamma_{yz}, \gamma_{zx})$ are required to be continuous functions of the coordinates(x,y,z), they are considered to be compatible. Thus, when a given body strained, the resulting displacements produce no gaps between small, adjacent elements of the body. The following six compatibility equations that need to be satisfied are obtained by various combinations of the partial derivatives of the six strains (3.12), with respect to the coordinates.
The static equilibrium equation can be changed to displacement terms as shown in the next equation.

$$\begin{aligned}
&\frac{\partial^2 \varepsilon_x}{\partial y^2} + \frac{\partial^2 \varepsilon_y}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} \\
&\frac{\partial^2 \varepsilon_y}{\partial z^2} + \frac{\partial^2 \varepsilon_z}{\partial y^2} = \frac{\partial^2 \gamma_{yz}}{\partial y \partial z} \\
&\frac{\partial^2 \varepsilon_z}{\partial x^2} + \frac{\partial^2 \varepsilon_x}{\partial z^2} = \frac{\partial^2 \gamma_{zx}}{\partial z \partial x} \\
&2\left(\frac{\partial^2 \varepsilon_x}{\partial y \partial z}\right) = \frac{\partial}{\partial x}\left(-\frac{\partial \gamma_{yz}}{\partial x} + \frac{\partial \gamma_{zx}}{\partial y} + \frac{\partial \gamma_{xy}}{\partial z}\right) \\
&2\left(\frac{\partial^2 \varepsilon_y}{\partial z \partial x}\right) = \frac{\partial}{\partial y}\left(\frac{\partial \gamma_{yz}}{\partial x} - \frac{\partial \gamma_{zx}}{\partial y} + \frac{\partial \gamma_{xy}}{\partial z}\right) \\
&2\left(\frac{\partial^2 \varepsilon_z}{\partial x \partial y}\right) = \frac{\partial}{\partial z}\left(\frac{\partial \gamma_{yz}}{\partial x} + \frac{\partial \gamma_{zx}}{\partial y} - \frac{\partial \gamma_{xy}}{\partial z}\right)
\end{aligned} \tag{3.15}$$

Using Hook's law, the compatibility conditions may be expressed in terms of stress components. The results, known as the Beltrami-Michell equations, are as follows:

$$
\begin{aligned}
\nabla^2\sigma_x + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial x^2} &= \frac{-\nu}{1-\nu}\left(\frac{\partial B_x}{\partial x}+\frac{\partial B_y}{\partial y}+\frac{\partial B_z}{\partial z}\right)-2\frac{\partial B_x}{\partial x}\\
\nabla^2\sigma_y + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial y^2} &= \frac{-\nu}{1-\nu}\left(\frac{\partial B_x}{\partial x}+\frac{\partial B_y}{\partial y}+\frac{\partial B_z}{\partial z}\right)-2\frac{\partial B_y}{\partial y}\\
\nabla^2\sigma_z + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial z^2} &= \frac{-\nu}{1-\nu}\left(\frac{\partial B_x}{\partial x}+\frac{\partial B_y}{\partial y}+\frac{\partial B_z}{\partial z}\right)-2\frac{\partial B_z}{\partial z}\\
\nabla^2\tau_{yz} + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial y\partial z} &= -\left(\frac{\partial B_y}{\partial z}+\frac{\partial B_z}{\partial y}\right)\\
\nabla^2\tau_{zx} + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial z\partial x} &= -\left(\frac{\partial B_z}{\partial x}+\frac{\partial B_x}{\partial z}\right)\\
\nabla^2\tau_{xy} + \frac{1}{1+\nu}\frac{\partial^2 I_1}{\partial x\partial y} &= -\left(\frac{\partial B_x}{\partial y}+\frac{\partial B_y}{\partial x}\right)
\end{aligned}
\qquad (3.16)
$$

where $I_1 = \sigma_x + \sigma_y + \sigma_z$ is the first stress invariant.

Returning to the static equilibrium equations for stress, a set of three differential equations can be derived for the three components of displacement by using Hook's law and compatibility relations. The result is

$$
\begin{aligned}
\nabla^2 u + \frac{1}{1-2\nu}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+\frac{B_x}{\mu} &= 0\\
\nabla^2 v + \frac{1}{1-2\nu}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+\frac{B_y}{\mu} &= 0\\
\nabla^2 w + \frac{1}{1-2\nu}\frac{\partial}{\partial z}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+\frac{B_z}{\mu} &= 0
\end{aligned}
\qquad (3.17)
$$

The differential equations of the consolidation, as given by Biot, are

$$G\nabla^2 u + \frac{G}{1-2\nu}\frac{\partial \varepsilon_v}{\partial x} - \frac{\partial p}{\partial x} = 0,$$
$$G\nabla^2 v + \frac{G}{1-2\nu}\frac{\partial \varepsilon_v}{\partial y} - \frac{\partial p}{\partial y} = 0, \qquad (3.18)$$
$$G\nabla^2 w + \frac{G}{1-2\nu}\frac{\partial \varepsilon_v}{\partial z} - \frac{\partial p}{\partial z} = 0.$$

and

$$k\nabla p^2 = \frac{\partial \varepsilon_v}{\partial t} \qquad (3.19)$$

where $\varepsilon_v = \varepsilon_x + \varepsilon_y + \varepsilon_z$ = volume change, p = incremental pore water pressure, ν = Poisson's ratio, E = Young's modulus, and $G = \frac{E}{2(1+\nu)}$ is shear modulus. The Laplace differential operator $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$.

There are three equations with four unknowns u, v, w, p. In order to have a complete system, one more equation is needed. This requires introducing Darcy's law governing the flow of water in a porous medium. Once again considering an element cube of soils, V_x is the volume of water flowing per second and the unit area through the surface of this cube is perpendicular to the x axis. V_y and V_z can be defined by the volume of water flowing per second in y and z direction. According to Darcy's law, these three components of the rate of flow are related to water pressure by the relations

$$V_x = -k\frac{\partial p}{\partial x}, V_y = -k\frac{\partial p}{\partial y}, V_z = -k\frac{\partial p}{\partial z} \qquad (3.20)$$

The physical constant k is called the coefficient of permeability of the soil. On the other hand, if the water is assumed to be incompressible, the

rate of water content of an element of soil must be equal to the volume of water entering per second through the surface of the element, hence

$$\frac{\partial \theta}{\partial t} = -\frac{\partial V_x}{\partial x} - \frac{\partial V_y}{\partial y} - \frac{\partial V_z}{\partial z} \tag{3.21}$$

Combining equations (3.12), (3.20) and (3.21) we obtain

$$k\nabla^2 p = \alpha \frac{\partial \varepsilon_v}{\partial t} + \frac{1}{Q}\frac{\partial p}{\partial t} \tag{3.22}$$

The four differential equations (3.18) and (3.22) are the basic equations satisfied by the four unknowns u, v, w, and p.
The incremental pore water pressure p of standard series form is shown as follows[8]:

$$p = \frac{4}{\pi}\frac{a-a_i}{\alpha a} p_0 \left\{ \exp\left[-\left(\frac{\pi}{2h}\right)^2 ct \right] \sin\frac{\pi z}{2h} + \frac{1}{3}\exp\left[-\left(\frac{3\pi}{2h}\right)^2 ct \right] \sin\frac{3\pi z}{2h} + \ldots \right\} \tag{3.23}$$

3.3 Consolidation theory of vertical drain by Barron.[4]

An embankment construction by stages or with surcharging may require an acceleration of consolidation by vertical drains (sand, pack and prefablicated drain). The consolidation analysis is then based on vertical and axi-symmetrical pore water flow.
The consolidation process in axi-symmetrical state can be calculated with the methods presented by Barron [4] and Hansbo. [43] In these methods, linear stress-strain relationships and constant soil parameters are assumed in the deformation process, as well as constant stress during the consolidation process and constant vertical surface displacements throughout the drained area. The effect of secondary compression is also neglected. With these assumptions, the equation of vertical consolidation becomes:

$$c_v \frac{\partial^2 u}{\partial x^2} + c_h \left(\frac{1}{\rho} \frac{\partial u}{\partial \rho} + \frac{\partial^2 u}{\partial \rho^2} \right) = \frac{\partial u}{\partial t} \tag{3.24}$$

where

c_v = coefficient of vertical consolidation

c_h = coefficient of horizontal consolidation

u = excess pore pressure
x = vertical space coordinate
ρ = radial space coordinate
t = time.

If the degree of consolidation U is defined as the degree of dissipation of excess pore pressure, it can, according to Carillo[14], be expressed as

$$U = U_h + U_v - U_h U_v \tag{3.25}$$

where
U = degree of total consolidation

U_h = degree of horizontal consolidation

U_v = degree of vertical consolidation.

The degree of vertical consolidation U_v is calculated according to Tezaghi's theory and the degree of horizontal consolidation U_h is calculated from Barron.[4]

$$U_h = 1 - \exp(-8T_h / \mu) \tag{3.26}$$

where T_h = time factor

$$T_h = c_h t / D^2 \tag{3.27}$$

$$\mu = \frac{n^2}{n^2 - 1}\left(\ln n - \frac{3}{4} + \frac{1}{n^2} - \frac{1}{4n^2}\right) \tag{3.28}$$

$$n = \frac{D}{d} \tag{3.29}$$

where
D = diameter of dewatered soil cylinder
d = drain diameter.

The vertical consolidation method presented in this section can be applied to sand drains and other vertical drains.

3.4 The governing equations for multiphase flow in a deforming porous medium.[52,79]

In this section the governing equations for multiphase flow in a deforming porous medium are developed. The general case of miscible three-phase flow is considered, including the saturated-unsaturated flow and one-phase flow (consolidation) problems.
The governing equations are developed on the lines of Biot's self-consistent theory.[8,9,10,11] The solid phase is assumed to comprise a porous skeleton of particles surrounded by one or more fluids (liquid and gaseous hydrocarbons and water). The shear stresses in the fluid phase are small whilst an all-around pressure is exerted on the solid phase. The small strain theory is considered to be applicable throughout this thesis, and Darcy's law is assumed valid in terms of the relative velocity of the fluid to grains, although it has no bearing on small rock matrix deformations.

3.4.1. Solid-phase behaviour.[52]

It is assumed that a pure (external and internal) fluid pressure **p** causes

only a uniform, volumetric strain by compressing the grains and that the major deformation of the porous skeleton is governed by the effective stress vector σ'. This is defined as follows, indicating tension is positive,

$$\boldsymbol{\sigma} = \sigma' - \mathbf{m}p \tag{3.30}$$

where

$\boldsymbol{\sigma}$ are total stresses as $\{\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\}^T$

σ' are effective stresses as $\{\sigma'_x, \sigma'_y, \sigma'_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\}^T$

$\mathbf{m} = \{1,1,1,0,0,0\}^T$ and

p = pore pressure.

Equation (3.30) can be obtained by pure statics which allows the total stress vector to be split into convenient super-imposable parts. This equation is also known as the effective stress principle first formulated by Terzaghi[88] (see Figure 3.2). The precise meaning of p in a region where multiphase flow exists is discussed in the next section.

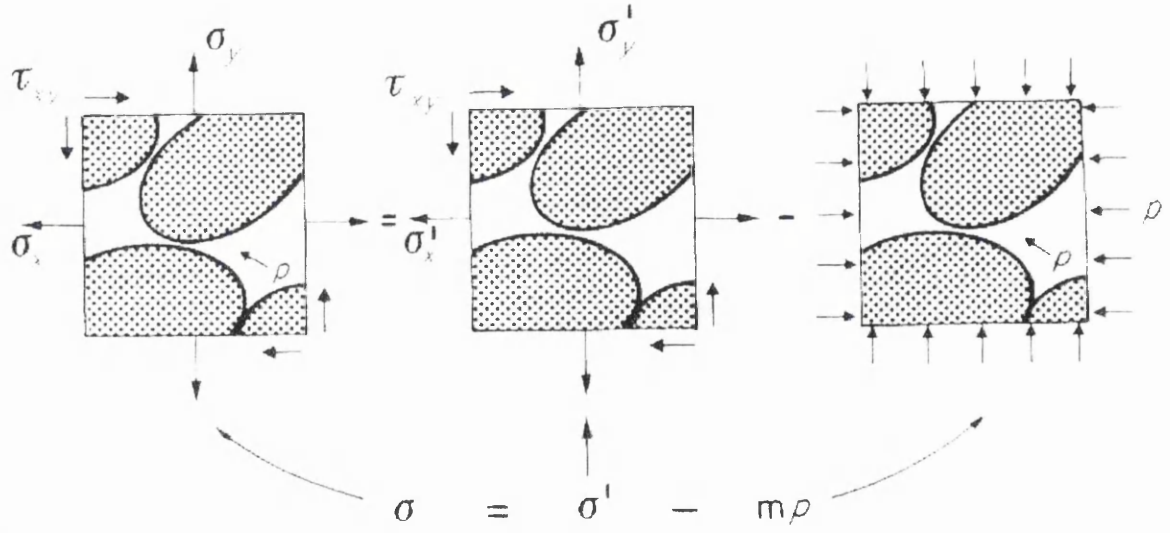


Figure 3.2 Total and effective stress in porous media

The constitutive equation relating this effective stress to the strains of

the skeleton is now independent of the pore pressure p, and for a general non-linear material can be written in a tangential form, thus allowing plasticity to be incorporated. If creep strains are present, the expression is written in general form as

$$d\boldsymbol{\sigma}' = \mathbf{D}_T(d\boldsymbol{\varepsilon} - d\boldsymbol{\varepsilon}_c - d\boldsymbol{\varepsilon}_p - d\boldsymbol{\varepsilon}_0) \tag{3.31}$$

where $d\boldsymbol{\varepsilon}$ represents the total strain of the skeleton,

$$d\boldsymbol{\varepsilon}_c = c\,dt \tag{3.32}$$

and creep strain is

$$d\boldsymbol{\varepsilon}_p = -\mathbf{m}(dp/3K_s) \tag{3.33}$$

Equation (3.33) represents the overall volumetric strains caused by uniform compression of the particles (as opposed to skeleton) by the pressure of the pore fluid, with K_s being the bulk modulus of the solid phase. In soils the latter volumetric strain is relatively insignificant and can be ignored, but it is important in rock mechanics where the compressibility of the solid phase is comparable to that of the skeleton. Finally $\boldsymbol{\varepsilon}_0$ represents all other strains not directly associated with stress changes (swelling, thermal, chemical etc.) i.e. the 'autogeneous' strains.[93]

The matrix $\mathbf{D}_T$ and the creep function c are dependent on the level of effective stresses $\boldsymbol{\sigma}'$ and also, if strain effects are considered, on the total strain of the skeleton $\boldsymbol{\varepsilon}$. Various constitutive models defining the tangent matrix $\mathbf{D}_T$ will be discussed later.

The equilibrium equation relating the total stress $\boldsymbol{\sigma}$ to the body forces $\mathbf{b}$ and the boundary traction $\hat{\mathbf{t}}$ specified at the boundary Γ of the domain Ω is formulated in terms of the unknown displacement vector $\mathbf{u}$. Using the principle of virtual work, the general equilibrium statement can be written as

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\boldsymbol{\sigma}d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}\mathbf{b}d\Omega-\int_{\Gamma}\delta\mathbf{u}^{T}\hat{\mathbf{t}}d\Gamma=0 \tag{3.34}$$

for virtual displacements $\delta\mathbf{u}$ such as that on the boundary part Γ_u where displacements are prescribed, these are not varied. Equation (3.34) is already a weak statement of the equilibrium relationship which also incorporates the boundary conditions.

The same form can be obtained, starting from the equilibrium equations, by eliminating the second derivatives via Green's transformation and solving the boundary value problem.[92]

The equilibrium statement (3.34) is also valid in incremental form,

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}d\boldsymbol{\sigma}d\Omega-\int_{\Omega}\delta\mathbf{u}^{T}d\mathbf{b}d\Omega-\int_{\Gamma}\delta\mathbf{u}^{T}d\hat{\mathbf{t}}d\Gamma=0 \tag{3.35}$$

If the effective stress relationship is incorporated into this equation, the following equation is obtained:

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}d\boldsymbol{\sigma}'d\Omega-\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{m}dpd\Omega-d\hat{\mathbf{f}}=0 \tag{3.36}$$

where

$$d\hat{\mathbf{f}}=\int_{\Omega}\delta\mathbf{u}^{T}d\mathbf{b}d\Omega+\int_{\Gamma}\delta\mathbf{u}^{T}d\hat{\mathbf{t}}d\Gamma \tag{3.37}$$

Equation (3.37) represents the change in external force due to boundary and body force loadings.

Further, on taking into account the constitutive relationship given by

equation (3.32), (3.33) and (3.35) and dividing by dt, the following equation is obtained:

$$\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{D}_{T}\frac{\partial}{\partial t}\boldsymbol{\varepsilon}d\Omega-\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{m}\frac{\partial p}{\partial t}d\Omega+\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{D}_{T}\mathbf{m}\frac{\partial p}{\partial t}\frac{1}{3K_{s}}d\Omega$$
$$-\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{D}_{T}\mathbf{c}d\Omega-\int_{\Omega}\delta\boldsymbol{\varepsilon}^{T}\mathbf{D}_{T}\frac{\partial}{\partial t}\boldsymbol{\varepsilon}_{0}d\Omega-\frac{\partial}{\partial t}\hat{\mathbf{f}}=0 \tag{3.38}$$

If the pressure p which in the case of a multiphase flow regime is an average value is known, then the displacements and stresses within the system can be obtained from this equation. However, the pressure field is generally coupled with the strain changes and these appear explicitly in the following flow equations.

3.4.2 Fluid phase behaviour in a deforming porous medium.

The geometrical complexity of a porous medium renders impossible a strict analytical treatment of the fluid velocity within the pore space. To overcome this obstacle, the fictitious seepage velocity (also known as bulk or Darcy velocity) is defined as

$$q=-\frac{1}{\mu}\mathbf{k}\nabla(p+\rho gh) \tag{3.39}$$

where

$\mathbf{k}=\begin{bmatrix}k_x & 0 & 0\\ 0 & k_y & 0\\ 0 & 0 & k_z\end{bmatrix}$ is the absolute permeability matrix of the medium,

μ is the dynamic viscosity of the fluid,
p is the fluid pressure,
ρ is the density,
g is the gravity,
and
h is the head above some arbitrary datum.[29]

Consider the case of a porous medium in which three phases, viz. liquid hydrocarbons, gaseous hydrocarbons and liquid water, are flowing simultaneously. On mathematically modelling such a system, the behaviour of a particular fluid is changed due to the interference of the other fluids in the pore space.
The volume fraction of the pore space occupied by one fluid is defined as the saturation of that phase and is denoted by S_π for the π phase. It was widely assumed that the permeability of each phase was proportional to its saturation at that point. However experimental studies have shown that multiphase flow reduces the permeability of the medium to the mixture as well as to the fluids individually.[67] Therefore, a relative permeability function $k_{r\pi}(S_\pi), k_{r\pi}(S_\pi)$ needs to be introduced to modify the permeability matrix $\mathbf{k}$. Although analytical expressions for these functions do exist,[23,51] their values in practice are usually obtained by experiment.[13,77]

Typical curves for the relative permeability in a two-phase flow are shown in Figure 3.3, where n denotes the non-wetting phase and w the wetting phase. The saturation of either phase is obtained from the capillary pressure versus saturation curve as shown in Figure 3.4. This type of relationship may be obtained by experiment on the particular porous medium under consideration.[90]
In the case of a three-phase flow problem, the functions are developed from two-phase data,[25] which consists of a set of oil/water relative permeability and oil/gas relative permeability data. From the two sets of data the values of k_{rg}, k_{rw} and k_{r0} are determined. An example of a composite three-phase curve is shown in Figure 3.5.

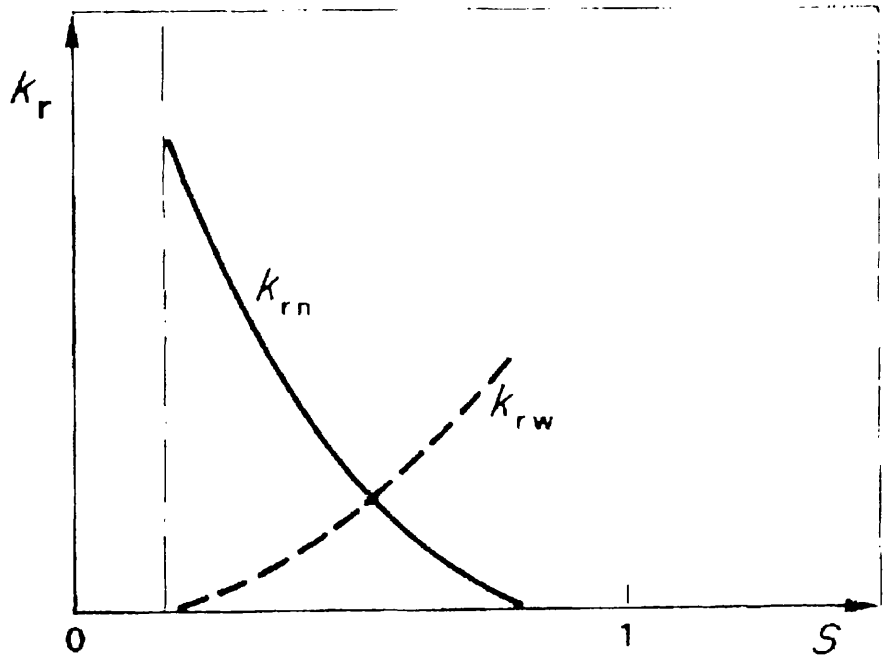


Figure 3.3 Typical permeability curves for oil and water

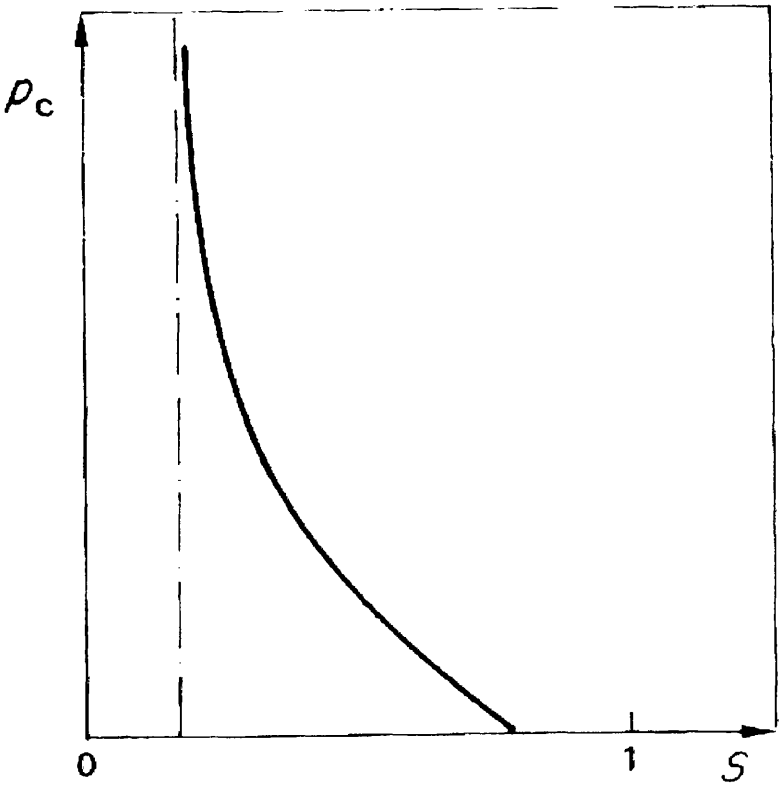


Figure 3.4 Typical saturation-capillary pressure curves

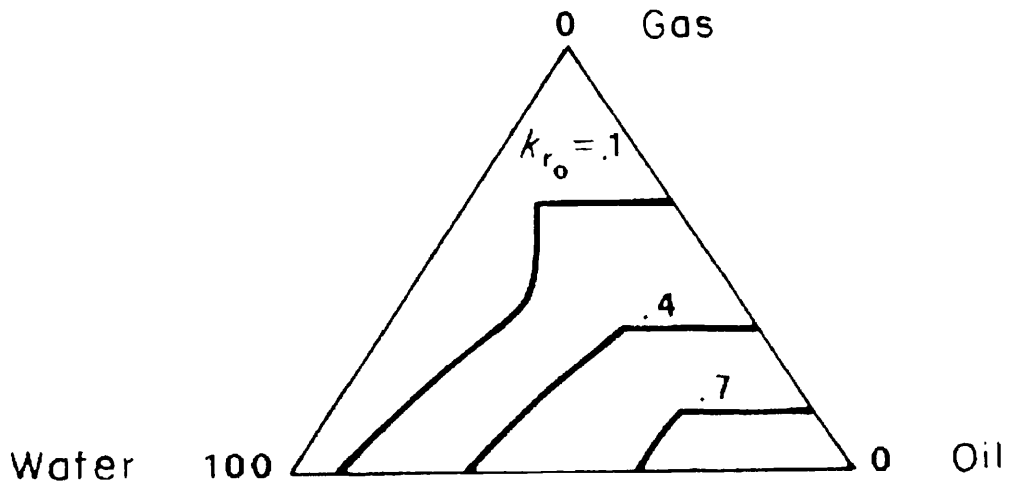


Figure 3.5 Composite three-phase curve

When one-phase flow only is considered, the continuity of flow requires that the following expression is valid:[25]

$$\text{(rate of fluid accumulation)} + \nabla\cdot(\rho q) = 0 \tag{3.40}$$

which, on combining with Darcy's law, results in

$$\text{(rate of fluid accumulation)} + \nabla\cdot\left(-\frac{k\rho}{\mu}\nabla(p+\rho gh)\right) = 0 \tag{3.41}$$

When liquid hydrocarbons, in both liquid and gaseous states, and water are flowing together, the oil and water are the only phases which can be regarded as immiscible. The solution of gas in either of the liquid phases brings about changes in the behaviour and physical properties of oil and water. These phenomena are handled by defining two parameters.

Firstly the formation volume factor $B_\pi(p_\pi)$ relates the volume of a unit mass of the π-phase at a pressure p_π to its volume at standard or stock tank conditions, i.e., 15.56°C (60°F) at standard atmospheric pressure.[22] For gaseous hydrocarbons the reference volume is the volume at surface conditions. The definition of B_π differs according to whether π is a

wetting or a non-wetting phase. If π is non-wetting, then

$$B_\pi(p_\pi) = \frac{V_\pi(p_\pi)}{V_{\pi STC}} \tag{3.42}$$

where V_π is the volume of a unit mass of the π-phase at pressure p_π, and $V_{\pi STC}$ is the volume of unit mass of the π-phase at stock tank conditions (STC).
In a wetting situation the phase has two components, a wetting fluid and a dissolved non-wetting component. In this case

$$B_\pi(p_\pi) = \frac{V_\pi(p_\pi) + \overline{V}_\pi(p_\pi)}{V_{\pi STC}} \tag{3.43}$$

where $\overline{V}_\pi$ is the volume of the dissolved non-wetting π-fluid at pressure P_π. This definition requires that $\overline{V}_{\pi STC} = 0$, i.e. the solubility is zero at STC. It is evident that the formation volume factor relates the density change with pressure.
Secondly, the solution ratio is used to account for the solubility of gas in the oil and water phases. Gaseous hydrocarbons occur in the system in three forms, free gas, gas dissolved in oil; gas dissolved in water.

In order to calculate the correct volume of dissolved gas, the solution ratio $R_{S\pi}$ is defined by

$$R_{S\pi} = \frac{V_{dgSTC}}{V_{\pi STC}} \tag{3.44}$$

where V_{dgSTC} is the volume that the gas dissolved in unit mass of the π-

phase would occupy at STC, and $V_{\pi STC}$ is the volume of the π-phase at STC.[22]

Incorporating these parameters in a three-phase system, we can express the continuity equation for oil in the following form,

$$\nabla^T\left\{\mathbf{k}\frac{k_{ro}(S_o)\rho_o}{\mu_o B_o(p_o)}\nabla(p_o+\rho_o gh)\right\}=\text{(rate of fluid accumulation)} \tag{3.45}$$

There are many factors which contribute to the rate of fluid accumulation and these are enumerated as follows:[54]

(a) Rate of change of total strain

$$\frac{\partial \varepsilon_v}{\partial t}=\mathbf{m}^{\mathrm{T}}\frac{\partial \boldsymbol{\varepsilon}}{\partial t} \tag{3.46}$$

(b) Rate of change of the grain volume due to pressure changes

$$\frac{(1-\phi)}{K_s}\frac{\partial \bar{p}}{\partial t} \tag{3.47}$$

where ϕ is the porosity and $\bar{p}$ is the average effective pressure. It will be shown later that $\bar{p}$ is a function of the saturation S_π of the three phases, i.e.

$$\bar{p}=S_o p_o+S_w p_w+S_g p_g \tag{3.48}$$

(c) Rate of change of saturation

$$\phi\frac{\rho_o}{B_o}\frac{\partial S_o}{\partial t} \tag{3.49}$$

(d) Rate of change of fluid density

$$\phi S_o \frac{\partial}{\partial t}\left(\frac{\rho_o}{B_o}\right) \tag{3.50}$$

(e) And finally the change of grain size due to effective stress changes $\partial\{\sigma'\}/\partial t$

$$-\frac{1}{3K_s}\mathbf{m}^T\mathbf{D}_T\left(\frac{\partial}{\partial t}\boldsymbol{\varepsilon}+\frac{\mathbf{m}}{3K_s}\frac{\partial \bar{p}}{\partial t}-\mathbf{c}\right) \tag{3.51}$$

Substituting for $\partial\boldsymbol{\sigma}'/\partial t$ from equation (3.33) yields the following expression:

$$-\frac{1}{3K_s}\mathbf{m}\mathbf{D}_T\left(\frac{\partial}{\partial t}\boldsymbol{\varepsilon}+\frac{\mathbf{m}}{3K_s}\frac{\partial \bar{p}}{\partial t}-\mathbf{c}\right) \tag{3.52}$$

The continuity equation for oil (with no source terms) becomes:

$$\begin{aligned}&-\nabla^T\left\{\mathbf{k}\frac{k_{ro}\rho_o}{\mu_o B_o}\nabla(p_o+\rho_o gh)\right\}+\phi\frac{\rho_o}{B_o}\frac{\partial S_o}{\partial t}+\phi S_o\frac{\partial}{\partial t}\left(\frac{\rho_o}{B_o}\right)+\rho_o\frac{S_o}{B_o}\\&\times\left\{(\mathbf{m}^T-\frac{\mathbf{m}^T\mathbf{D}_T}{3K_S})\frac{\partial\boldsymbol{\varepsilon}}{\partial t}+\frac{\mathbf{m}^T\mathbf{D}_T\mathbf{c}}{3K_S}+\left[\frac{(1-\phi)}{K_S}-\frac{1}{(3K_S)^2}\mathbf{m}^T\mathbf{D}_T\mathbf{m}\right]\frac{\partial\bar{p}}{\partial t}\right\}=0\end{aligned} \tag{3.53}$$

The continuity equation for water is obtained in a similar way:

$$\begin{aligned}&-\nabla^T\left\{\mathbf{k}\frac{k_{rw}\rho_w}{\mu_w B_w}\nabla(p_w+\rho_w gh)\right\}+\phi\frac{\rho_w}{B_w}\frac{\partial S_w}{\partial t}+\phi S_w\frac{\partial}{\partial t}\left(\frac{\rho_w}{B_w}\right)\\&+\rho_w\frac{S_w}{B_w}\{(\mathbf{m}^T-\frac{\mathbf{m}^T\mathbf{D}_T}{3K_S})\frac{\partial\boldsymbol{\varepsilon}}{\partial t}+\frac{\mathbf{m}^T\mathbf{D}_T\mathbf{c}}{3K_S}\\&+\left[\frac{(1-\phi)}{K_S}-\frac{1}{(3K_S)^2}\mathbf{m}^T\mathbf{D}_T\mathbf{m}\right]\frac{\partial\bar{p}}{\partial t}\}=0\end{aligned} \tag{3.54}$$

The continuity equation for gas, taking into account the possible sources of gas through the solution ratio $R_{R\pi}$, is obtained as follows:

$$
\begin{aligned}
&-\nabla^{T}\{\mathbf{k}\rho_g\left[\frac{k_{r\pi}}{\mu_g B_g}+\frac{R_{so}k_{ro}}{\mu_o B_o}+\frac{R_{sw}k_{rw}}{\mu_w B_w}\right]\nabla(p_g+\rho_g gh)\\
&+\phi\frac{\partial}{\partial t}\left\{\rho_g\left[\frac{S_g}{B_g}+\frac{R_{so}S_o}{B_o}+\frac{R_{sw}S_w}{B_w}\right]\right\}\\
&+\rho_g\left[\frac{S_g}{B_g}+\frac{R_{so}S_o}{B_o}+\frac{R_{sw}S_w}{B_w}\right]\{\left(\mathbf{m}^{T}-\frac{\mathbf{m}^{T}\mathbf{D}_T}{3K_S}\right)\frac{\partial\boldsymbol{\varepsilon}}{\partial t}+\frac{\mathbf{m}^{T}\mathbf{D}_T\mathbf{c}}{3K_S}\\
&+\left[\frac{1-\phi}{K_S}-\frac{1}{(3K_S)^2}\mathbf{m}^{T}\mathbf{D}_T\mathbf{m}\right]\frac{\partial\bar{p}}{\partial t}\}=0
\end{aligned}
\tag{3.55}
$$

In equations (3.53)-(3.55), $k_{r\pi}$ is a function of the saturation of the π-phase, and B_π a function of p_π. The three continuity equations are subjected to the condition that

$$S_o+S_w+S_g=1 \tag{3.56}$$

and need to be solved simultaneously with equation (3.38) and the expression involving $\bar{p}$. This will be discussed in the next section.

An additional control on the stability and accuracy of the solution is the material balance check, because in order for a model to maintain stability, it should conserve mass at all times.[25] The material balance equation will be dealt with in detail in one of the solution procedures discussed later. The equations governing the immiscible two-phase flow regimes in a deforming porous medium can be easily obtained from the previously stated general equations.

3.4.3 The fluid pressure in a multiphase flow regime.

To this point, the governing equations have been derived directly at the macroscopic level. But to deal correctly with the fluid pressures in the case of multiphase flow occurring in the pore space, microscopic level must be considered. In this section, attention is focused on what happens at a mathematical point within a single phase. At this level the porous

medium is a heterogeneous medium due to the presence of the solid-fluid phase interfaces.

A general description, as above, of the processes taking place in a porous medium is of no practical value because of the complexity of the domain and the impossibility of measurements of physical quantities at this level. Hence, for the governing equations of the porous medium, a continuum approach has been adopted, in which each phase present in the system is assumed to fill up the entire porous medium domain, forming an overlapping continuum.[7] At each point in the domain and for every phase, including the solid phase, the variables taken into account in the previous sections are therefore definable. The variables are in fact average values over a representative elementary volume (REV)[7] around any considered point in the porous medium domain. This representative elementary volume is defined in such a way that wherever it is placed within the considered porous medium domain, it always contains both the solid phase and void space. Furthermore it is assumed that within the REV both the solid skeleton and the void space are more or less evenly distributed.
Whitaker[89] has shown that valid results are obtained with the averaging process if

$$d \ll l \ll L \tag{3.57}$$

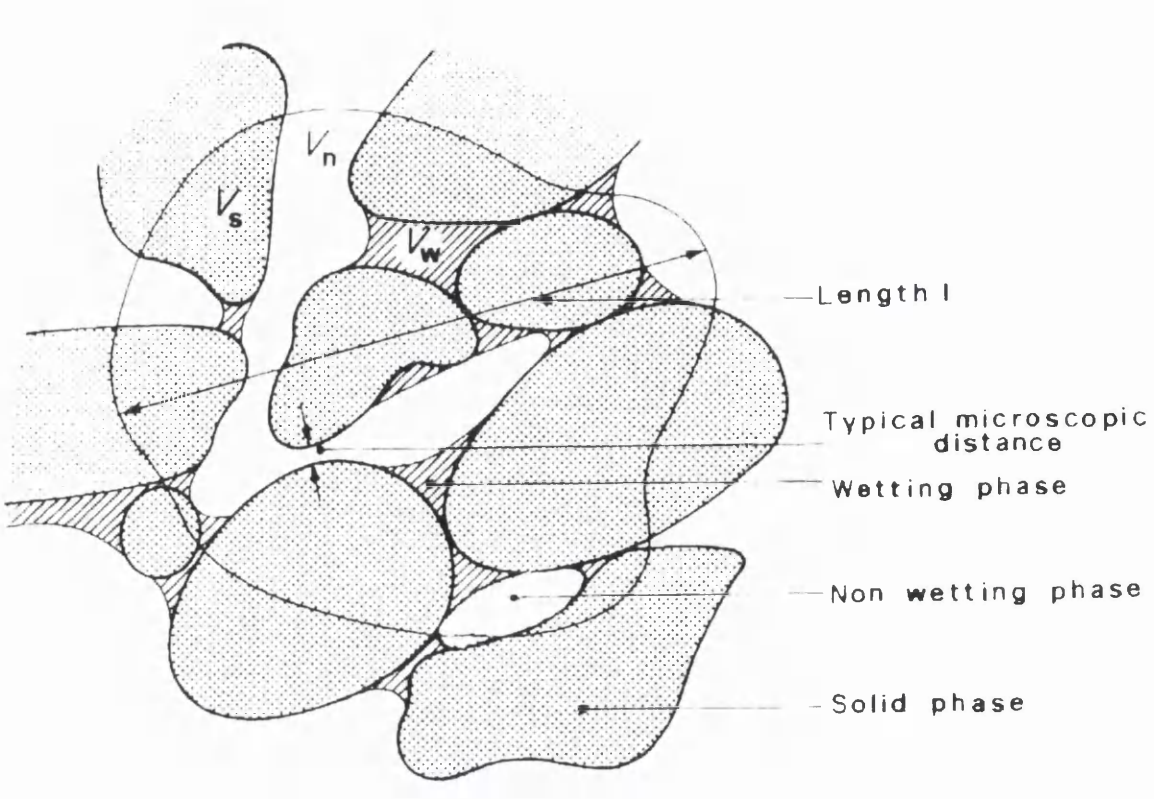


Figure 3.6 Typical averaging volume(two-phase flow)

where d is a characteristic microscopic length over which significant variations in the space dependent quantities occur, l is a characteristic length of the averaging volume and L is a characteristic macroscopic length of the porous medium domain, over which significant changes in average (macroscopic) quantities occur (see Figure 3.6)

The requirement (3.57) ensures that the size selected for the REV will remove the effect of the microscopic in-homogeneity without eliminating the effect of macroscopic in-homogeneity.

As long as the REV is independent of both time and location in the porous medium and satisfies the requirement listed above, the averaged equations obtained are independent of the geometry of the REV. The volume V of a REV is composed by the sum of the volume V_π of the single phases π

$$V = \sum_{\pi} V_\pi \tag{3.58}$$

where the subregion of a REV occupied by the solid component is treated as a phase in its own right with a volume V_s. Additionally, the region V_f of a REV is assigned to the region occupied by the fluids. The phase average of some quantity ζ in the π-phase is defined by the expression

$$\langle \zeta_\pi \rangle = \frac{1}{V} \int_V \zeta_\pi dV \tag{3.59}$$

where ζ_π is the value of ζ within the π-phase and is taken to be zero in all other phases.

It is worth noting that, according to the assumption of an overlapping continuum, $<\zeta_\pi>$ is an average associated with a given point which may not lie within the π-phase, but its value may still be non-zero even if that point lies inside the π'-phase, where $\pi' \neq \pi$.

If the volume in equation (3.58) is replaced by the volume of the π-phase V_π, we have an intrinsic phase average

$$\langle \zeta_\pi \rangle^\pi = \frac{1}{V_\pi} \int_{V_\pi} \zeta_\pi dV \tag{3.60}$$

Since ζ_π is zero in all phase $\pi' \neq \pi$

$$\langle \zeta_\pi \rangle^\pi = \frac{1}{V_\pi} \int_V \zeta_\pi dV = \frac{1}{V_\pi} \langle \zeta_\pi \rangle \tag{3.61}$$

Next, the pore space occupied by three fluid phases as oil, water and gas will be considered.

The (averaged) total stress vector at macroscopic level may be expressed in terms of intrinsic phase averages as follows:

$$\begin{aligned}\langle\boldsymbol{\sigma}\rangle &= \frac{1}{V}\int_V \boldsymbol{\sigma}\,dV = \frac{1}{V}\left[\int_{V_s}\boldsymbol{\sigma}\,dV + \int_{V_f}\boldsymbol{\sigma}\,dV\right] \\ &= \frac{V_s}{V}\langle\boldsymbol{\sigma}_s\rangle^s + \frac{V_f}{V}\left[\frac{V_o}{V_f}\langle\boldsymbol{\sigma}_o\rangle^o + \frac{V_w}{V_f}\langle\boldsymbol{\sigma}_w\rangle^w + \frac{V_g}{V_f}\langle\boldsymbol{\sigma}_g\rangle^g\right] \\ &= (1-\phi)\langle\boldsymbol{\sigma}_s\rangle^s + \phi\left[S_o\langle\boldsymbol{\sigma}_o\rangle^o + S_w\langle\boldsymbol{\sigma}_w\rangle^w + S_g\langle\boldsymbol{\sigma}_g\rangle^g\right]\end{aligned} \tag{3.62}$$

For each fluid phase $\sigma_\pi = \tau_\pi - \mathbf{m}p_\pi$, where τ_π denotes the shear stress vector, p_π denotes the pressure (positive for compression) and $\mathbf{m} = [1,1,1,0,0,0]^T$ as already indicated in Section 3.4.1. It is assumed that τ_π is negligible in a fluid phase.[6]

The total stress is then expressed by

$$\langle\boldsymbol{\sigma}\rangle = (1-\phi)\langle\boldsymbol{\sigma}_s\rangle^s - \phi\mathbf{m}\left[S_o\langle p_o\rangle^o + S_w\langle p_w\rangle^w + S_g\langle p_g\rangle^g\right] \tag{3.63}$$

For a single fluid phase filling the void space, the effective stress concept was given by equation (3.30). It should be remembered that each solid particle is assumed to be surrounded by the ambient fluid. Furthermore, the fluid pressure (in this case the common, macroscopic pressure) acting on the fluid solid interface of each grain causes only a uniform volumetric strain in the grain.

Equation (3.30) is now derived again in terms of intrinsic phase averages. For only one phase π flowing, equation (3.62) becomes

$$\begin{aligned}\langle\boldsymbol{\sigma}\rangle &= (1-\phi)\langle\sigma_s\rangle^s - \phi\mathbf{m}\langle p_\pi\rangle^\pi \\ &= (1-\phi)\left(\langle\sigma_s\rangle^s + \mathbf{m}\langle p_\pi\rangle^\pi\right) - (1-\phi)\mathbf{m}\langle p_\pi\rangle^\pi - \phi\mathbf{m}\langle p_\pi\rangle^\pi \\ &= \boldsymbol{\sigma}' - \mathbf{m}\langle p_\pi\rangle^\pi\end{aligned} \tag{3.64}$$

where

$$\boldsymbol{\sigma}' = (1-\phi)\left(\langle\sigma_s\rangle^s + \mathbf{m}\langle p_\pi\rangle^\pi\right) \tag{3.65}$$

is the strain-producing stress in the solid skeleton, the effective stress.[7]
According to Terzaghi's definition, the effective stress is the sum of the pressure and average stress in the solid phase.
The same concept can be extended to the case where more than one fluid phase occupies the pore space. It is necessary to replace as follows: $< p_\pi >^\pi$ of equation (3.65) with the average fluid pressure

$$\langle p_f \rangle^f = S_o \langle p_o \rangle^o + S_w \langle p_w \rangle^w + S_g \langle p_g \rangle^g \tag{3.66}$$

Omitting the averaging symbol, in order to deal with macroscopic level, equation (3.66) becomes

$$\bar{p} = S_o p_o + S_w p_w + S_g p_g \tag{3.67}$$

Weights other than the saturations are possible in equation (3.67), namely, the fractional areas of oil-solid, water-solid and gas-solid contact.

3.4.4 Flow equation for single phase in a deforming porous medium.

The case of water only, flowing at saturated conditions is considered here, further simplifying equation (3.54). Typical values for the formation volume factor for water can be found in the model developed by Lewis and Schrefler.[53] In the following analysis, the formation volume factor is assumed to be equal to unity. Furthermore, the water is assumed to behave isotropically with respect to the density ρ_w. Hence, dividing by ρ_w results in the equation (3.50) for water being written as

$$\phi \frac{1}{\rho_w} \frac{\partial \rho_w}{\partial t} = \phi \frac{1}{\rho_w} \frac{\partial \rho_w}{\partial \bar{p}} \frac{\partial \bar{p}}{\partial t} = \frac{\phi}{K_w} \frac{\partial \bar{p}}{\partial t} \tag{3.68}$$

where

$$\bar{p} = p_w \tag{3.69}$$

and K_w is the bulk modulus of water. Equation (3.54) then becomes

$$\begin{aligned} &-\nabla^T \left\{ \frac{\mathbf{k}}{\mu_w} \nabla(\rho_w + \rho_w g h) \right\} + \left(\mathbf{m}^T - \frac{\mathbf{m}^T \mathbf{D}_T}{3K_S} \right) \frac{\partial \boldsymbol{\varepsilon}}{\partial t} + \frac{\mathbf{m}^T \mathbf{D}_T \mathbf{c}}{3K_S} \\ &+ \left[\frac{(1-\phi)}{K_S} + \frac{\phi}{K_w} - \frac{1}{(3K_S)^2} \mathbf{m}^T \mathbf{D}_T \mathbf{m} \right] \frac{\partial p_w}{\partial t} = 0 \end{aligned} \tag{3.70}$$

Equation (3.70) was obtained by R. W. Lewis and others[53,55,91] and is usually written as

$$\begin{aligned} &-\nabla^T \left\{ \mathbf{K} \nabla \left(\frac{p_w}{\rho_w g} + h \right) \right\} + \left(\mathbf{m}^T - \frac{\mathbf{m}^T \mathbf{D}_T}{3K_S} \right) \frac{\partial \boldsymbol{\varepsilon}}{\partial t} + \frac{\mathbf{m}^T \mathbf{D}_T \mathbf{c}}{3K_S} \\ &+ \left[\frac{(1-\phi)}{K_S} + \frac{\phi}{K_w} - \frac{1}{(3K_S)^2} \mathbf{m}^T \mathbf{D}_T \mathbf{m} \right] \frac{\partial p_w}{\partial t} = 0 \end{aligned} \tag{3.71}$$

where

$$\mathbf{K} = \frac{\mathbf{k} p_w g}{\mu} \tag{3.72}$$

This is known as the permeability or hydraulic conductivity matrix, in which coefficients have units of length/time.

3.4.5 Saturated-unsaturated flow in porous media.

It will now be shown that equation (3.54) also describes the flow of water in an unsaturated or partly saturated porous medium.[79] After dividing by

the density ρ_w and taking the formation volume factor to be equal to unity, equation (3.54) in the region above the free surface $S_w<1$ becomes

$$-\nabla^T\left\{\mathbf{k}\frac{k_{rw}}{\mu_w}\nabla(p_w+\rho_w gh)\right\}+\phi\frac{\partial S_w}{\partial t}+\phi\frac{S_w}{\rho_w}\frac{\partial \rho_w}{\partial t}+S_w(\mathbf{m}^T-\frac{\mathbf{m}^T\mathbf{D}_T}{3K_S})\frac{\partial \boldsymbol{\varepsilon}}{\partial t}$$
$$+S_w\frac{\mathbf{m}^T\mathbf{D}_T\mathbf{c}}{3K_S}+S_w\left[\frac{1-\phi}{3K_S}-\frac{1}{(3K_S)^2}\mathbf{m}^T\mathbf{D}_T\mathbf{m}\right]\frac{\partial \bar{p}}{\partial t}\}=0 \qquad (3.73)$$

The pressure P_w and the relative permeability k_{rw} are functions of this saturation.[62]

These functional relationships between k_{rw}, p_w, S_w are different for each soil and are usually determined by field or laboratory experiments. Other examples of these relationships are given by Safai and Pinder. [71]

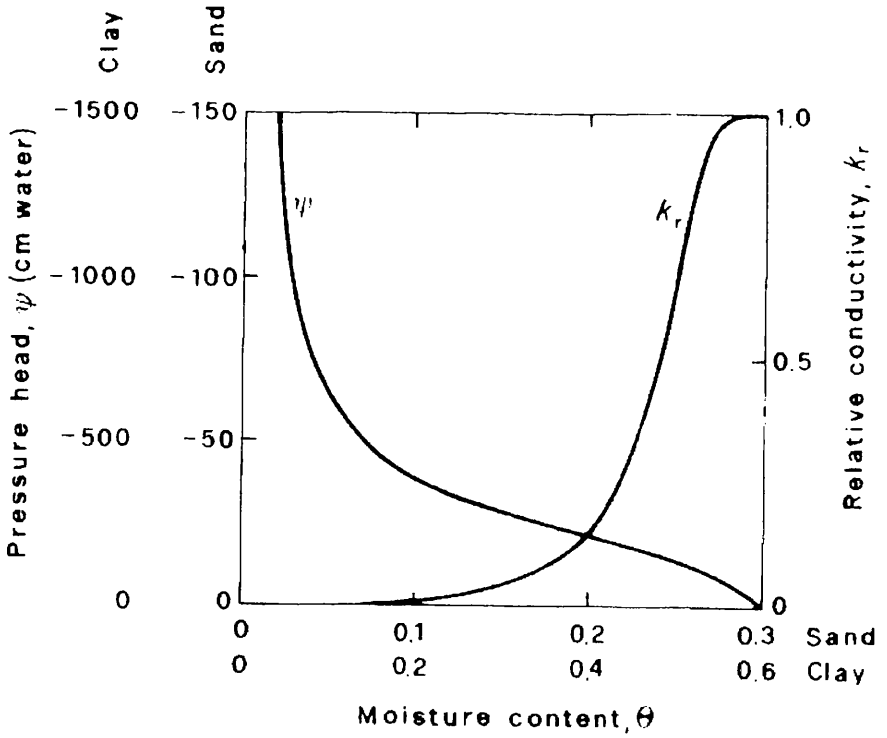


Figure 3.7 Typical unsaturated properties

Below the free surface the medium is fully saturated with $S_w=1$ and

equations (3.70) (3.71) and (3.72) hold. The pressure p_w is taken to be positive in the saturated zone and negative in the unsaturated zone.[62] For the sake of convenience, the following transformations are introduced in equation (3.75).

The rate of change of saturation, equation (3.51), can be written as

$$\phi \frac{\partial S_w}{\partial t} = \phi \frac{\partial S_w}{\partial p_w} \frac{\partial p_w}{\partial t} = C_s \frac{\partial p_w}{\partial t} \tag{3.74}$$

where C_s is the specific moisture content and is defined in terms of the pressure instead of the pressure head $\psi = p_w / \gamma_w$.[61,62] The partial derivative $\partial S_w / \partial p_w$ is obtained from the slope of the retention curve $S_w(\psi)$, as in Figure 3.7.

Equation (3.74), which represents an unsaturated medium, is not included in the constitutive equations of Safai and Pinder.[71]
On utilizing equation (3.68), the rate of change of fluid density, equation (3.50) becomes

$$\frac{\phi}{\rho_w} S_w \frac{\partial \rho_w}{\partial t} = \phi \frac{S_w}{K_w} \frac{\partial p_w}{\partial t} \tag{3.75}$$

and represents the ability of water to expand due to changes in hydrostatic pressure.

Substituting equation (3.74) and equation (3.75) into equation (3.73) yields

$$-\nabla^T\left\{\mathbf{k}\frac{k_{rw}}{\mu_w}\nabla(p_w+\rho_w gh)\right\}+S_w\left(\mathbf{m}^T-\frac{\mathbf{m}^T\mathbf{D}_T}{3K_S}\right)\frac{\partial\boldsymbol{\varepsilon}}{\partial t}+S_w\frac{\mathbf{m}^T\mathbf{D}_T\mathbf{c}}{3K_S}$$
$$+\bar{C}_s\frac{\partial\rho_w}{\partial t}+\phi\frac{S_w}{K_w}\frac{\partial\rho_w}{\partial t}+S_w\left[\frac{1-\phi}{3K_S}-\frac{1}{(3K_S)^2}\mathbf{m}^T\mathbf{D}_T\mathbf{m}\right]\frac{\partial\bar{p}}{\partial t}=0 \tag{3.76}$$

If it is assumed that the air phase is continuous in the unsaturated zone and remains at atmospheric pressure $p_{atm}=0$, [62,71] the average pressure $\bar{p}$ from equation (3.67) which appears in the last term of equation (3.76) becomes

$$\bar{p}=S_w p_w \tag{3.77}$$

And

$$\frac{\partial\bar{p}}{\partial t}=S_w\frac{\partial p_w}{\partial t}+p_w\frac{\partial S_w}{\partial t}=S_w\frac{\partial p_w}{\partial t}+\frac{\bar{C}_s}{\phi}p_w\frac{\partial p_w}{\partial t} \tag{3.78}$$

Some complications arise due to the last term of equation (3.78), where $p_w(\partial p_w/\partial t)$ appears. As stated previously, the effects of the rate of change of the grain volume due to pressure changes (3.47), as well as those of the change of grain size due to effective stress changes (3.51) are negligible in soil but have importance in the rock mechanics context.[91]

On neglecting the contributions of equations (3.47) and (3.51) the nonlinearity introduced through equation (3.78) disappears, but care has to be taken with the solid deformation equation (3.38). For the sake of completeness, these terms are included in the following equations.

Collecting the pressure terms, equation (3.73) becomes

$$-\nabla^{T}\left\{\mathbf{k}\frac{k_{rw}}{\mu_{w}}\nabla(p_{w}+\rho_{w}gh)\right\}+S_{w}\left(\mathbf{m}^{T}-\frac{\mathbf{m}^{T}\mathbf{D}_{T}}{3K_{S}}\right)\frac{\partial\boldsymbol{\varepsilon}}{\partial t}+S_{w}\frac{\mathbf{m}^{T}\mathbf{D}_{T}\mathbf{c}}{3K_{S}}$$
$$+\left[\bar{C}_{s}+\phi\frac{S_{w}}{K_{w}}+S_{w}\left(\frac{1-\phi}{K_{S}}-\frac{1}{(3K_{S})^{2}}\mathbf{m}^{T}\mathbf{D}_{T}\mathbf{m}\right)\left(S_{w}+\frac{\bar{C}_{s}}{\phi}p_{w}\right)\right]\frac{\partial p_{w}}{\partial t}=0 \qquad (3.79)$$

It should be noted that equation (3.79) obviously coincides with equations (3.77) (3.78) and (3.79) in the saturated zone.

It has been shown[79] that equation (3.79) contains the governing equation for saturated-unsaturated flow in porous media as given by Neuman.[61]

3.5 Concluding remarks.

In this chapter a general formulation of multiphase flow in a porous medium has been derived, which includes the cases of immiscible two-phase flow, one-phase flow (consolidation) and saturated-unsaturated flow.

The stage is now set for a description of the general numerical solution of these equations, and the presentation of particular finite element formulations which have been incorporated into computer programs.

Chapter 4

Constitutive relationships and variable permeability

Constitutive equations are important to solve the governing equation of continuum finite analysis for geotechnical or reservoir engineering problems. [52] They are necessary supplemented for solving the equilibrium and compability equations presented in Chapter 3, and provide a realistic representation of the stress-strain characteristics for porous medium. The choice of an appropriate constitutive relationship may have a significant influence on the numerical results obtained. In the case of a soil mass, the deformations will be associated with consolidation and with creep. In this context, creep is understood to refer to the deformations which occur when the effective stresses remain constant.

The constitutive relationship is expressed as a linear relation between small changes in strain, $\delta\varepsilon$ and small changes in effective stress $\{\delta\sigma'\}$.

$$\delta\varepsilon = D\delta\sigma' \tag{4.1}$$

The matrix D is usually symmetric, having a maximum of 21 independent components for a three-dimensional continuum. These components will generally depend on the accumulated strain ε and effective stress σ'. If the material is elastic and isotropic, the independent constants reduce to two.

Non-linear analysis of geotechnical problems generally uses the elastoplastic instead of non-linear elastic model because more accurate results are obtained. The elastoplastic model is used in this thesis to compare the developed program with the PLAXIS program. In determining the choice of a suitable soil constitutive model, a crucial point is to assign values to the constants defining the model. Determining the properties of clay soil requires complex experimentation to uncover

the soil properties that are appropriate for finite element analysis. The choice of these properties is itself a research field. [59] This is of particular importance in subsidence problems where lengthy histories are involved, and requires expensive trial runs for the assessment of each parameter. [80] An important objective is therefore to minimize the number of constants involved in the choice of constitutive law.

The state of stress in soil can be derived from two components: deviatoric and hydrostatic stress components. When dealing with isotropic models, it is more convenient to work in terms of stress invariants, which are defined in the next section.

4.1 Stress invariants

Stress invariants are stress combinations whose values are independent of the orientation of spatial reference axes. The principal stresses σ_1, σ_2 and σ_3, acting at right angles to each other, are themselves invariants.

The stress invariants used in the following are the mean stress p, the deviator stress q and the angular stress invariant θ. The definitions apply equally to effective stresses and total stresses.

$$p = -\frac{1}{3}(\sigma_x + \sigma_y + \sigma_z) \tag{4.2}$$

$$q^2 = \sigma_x(\sigma_x - \sigma_y) + \sigma_y(\sigma_y - \sigma_z) + \sigma_z(\sigma_z - \sigma_x) + 3(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2) \tag{4.3}$$

$$\theta = \frac{1}{3}\sin^{-1}\left[-\frac{27J_3}{2q^3}\right] \tag{4.4}$$

where

$$J_3 = \begin{vmatrix} \sigma_x + p & \tau_{xy} & \tau_{zx} \\ \tau_{xy} & \sigma_y + p & \tau_{yz} \\ \tau_{zx} & \tau_{yz} & \sigma_z + p \end{vmatrix} \tag{4.5}$$

The third stress invariant θ was introduced in 1972 by Nayak and Zienkiewicz. [58] Using this angular representation of the third stress

invariant leads to the following expression of the three principal stresses:

$$\begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\end{Bmatrix}=\begin{Bmatrix}p\\ p\\ p\end{Bmatrix}-\frac{2}{3}q\begin{Bmatrix}\sin(\theta_o-\frac{2}{3}\pi)\\ \sin\theta_o\\ \sin(\theta_o+\frac{2}{3}\pi)\end{Bmatrix} \tag{4.6}$$

where θ_0 is a part of the general solution to equation (4.4)

$$\theta=\theta_o+\frac{2}{3}\pi n, \tag{4.7}$$

and n = any integer which lies within the limits $-\pi/6\le\theta_0\le\pi/6$.

Inversely,

$$p=-\frac{1}{3}(\sigma_1+\sigma_2+\sigma_3) \tag{4.8}$$

$$q^2=\sigma_1(\sigma_1-\sigma_2)+\sigma_2(\sigma_2-\sigma_3)+\sigma_3(\sigma_3-\sigma_1) \tag{4.9}$$

$$\theta_o=\tan^{-1}\left[\frac{(\sigma_1-\sigma_2)-(\sigma_2-\sigma_3)}{\sqrt{3}(\sigma_1-\sigma_3)}\right] \qquad -\frac{\pi}{6}\le\theta_o\le\frac{\pi}{6} \tag{4.10}$$

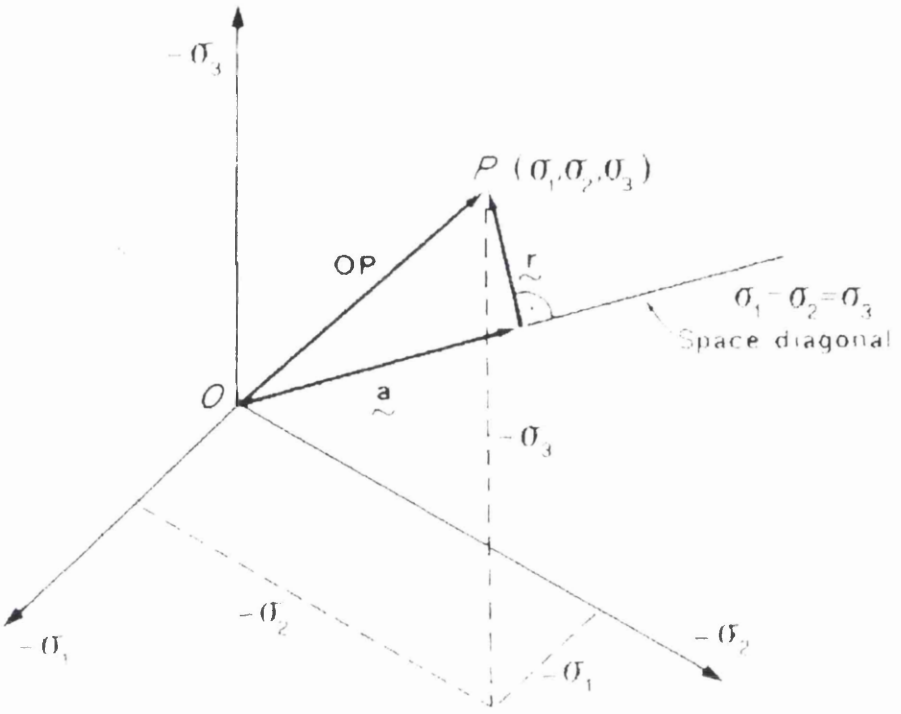


Figure 4.1 Stress point in the principal stress space

In a three-dimensional stress space with $\sigma_1,\sigma_2\sigma_3$ as orthogonal reference axes, a state of hydrostatic stress, given by $\sigma_1=\sigma_2=\sigma_3$, will be represented by a line passing through the origin equally inclined to each of the axes. This line is known as the space diagonal and its direction cosines are $1/\sqrt{3},1/\sqrt{3},1/\sqrt{3}$.

Point P (in Figure 4.1) represents an arbitrary stress state. With reference to the base axes, the vector $\overrightarrow{OP}$ has components $(-\sigma_1,-\sigma_2,-\sigma_3)$.

This stress vector can be decomposed into component a along the space diagonal and component r which lies in the π-plane and is perpendicular to the space diagonal (see Figure 4.2).

The projection of $\overrightarrow{OP}$ on the space diagonal gives the component a as

$$a=\frac{1}{\sqrt{3}}(-\sigma_1-\sigma_2-\sigma_3)=\sqrt{3}p. \tag{4.11}$$

The component r is given by

$$
\begin{aligned}
r^2 &= (OP)^2 - a^2 \\
&= (\sigma_1^2 + \sigma_2^2 + \sigma_3^2) - 3p^2 \\
&= (\sigma_1 + p)^2 + (\sigma_2 + p)^2 + (\sigma_3 + p)^2 \\
&= \frac{3}{2} q^2
\end{aligned}
\tag{4.12}
$$

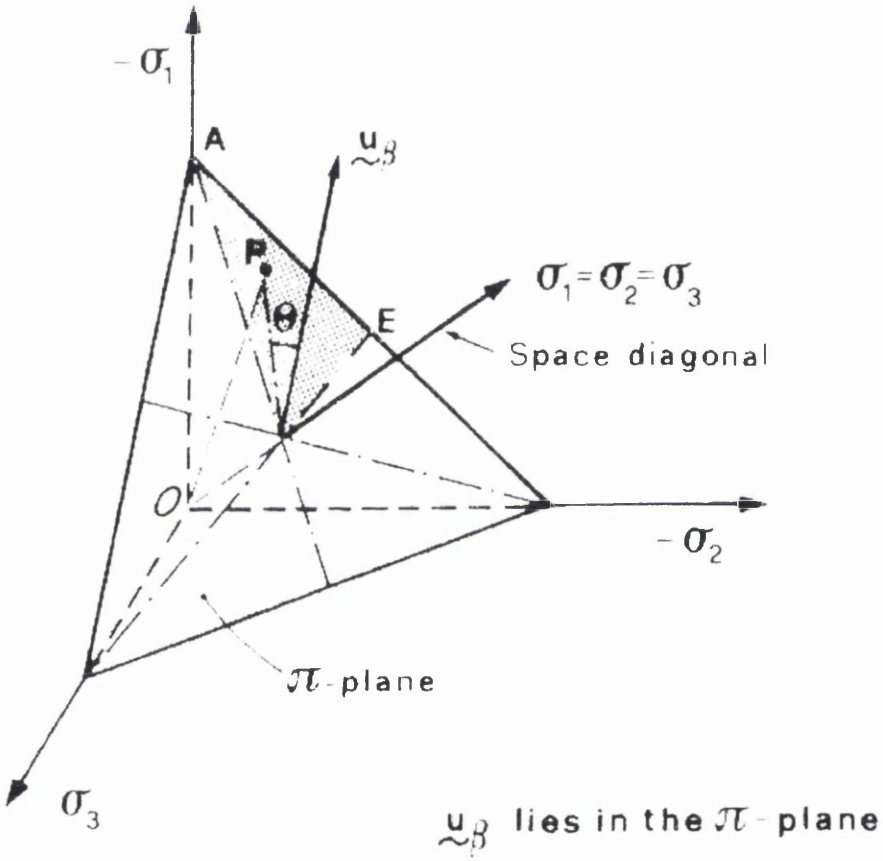


Figure 4.2 π – *plane*, hydrostatic and deviatoric component of the stress vector

In the principal stress space, therefore, a state of stress can be decomposed into a hydrostatic component a (=multiple of the first stress invariant) lying along the space diagonal, and a deviatoric component r (=multiple of the second stress invariant) lying in the π-plane. The deviatoric component ($\sqrt{3/2}\ q$ is the distance of the stress point P from the space diagonal. Because of the assumption of isotropy, stress functions, such as yield surfaces, are completely defined if taken over the sector 60 of π, where $\sigma_1 > \sigma_2 > \sigma_3$.

It can easily be shown that θ_0 is the orientation of r measured in the π-plane anti-clockwise from the bisector of APE (indicated as u_β in Figure 4.2). The geometric representation may be generalized to include the complete π-plane by replacing θ_0 with θ.

4.2 Linear elastic analyses.

The early development of numerical methods in geotechnical and reservoir engineering situations was based on the assumption of linear elasticity. In the case of consolidation analysis, a linear elastic behaviour of the soil skeleton is usually combined with linear fluid behaviour.

Isotropic linear elasticity requires the knowledge of two constants, which are usually the Young's modulus E and the Poisson ratio ν. In the case of a three-dimensional continuum, the $[D]$ matrix of equation (4.1) becomes

$$[D]=\frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ & 1-\nu & \nu & 0 & 0 & 0 \\ & & 1-\nu & 0 & 0 & 0 \\ & & & \frac{1-2\nu}{2} & 0 & 0 \\ & & & & \frac{1-2\nu}{2} & 0 \\ \textit{Symmetry} & & & & & \frac{1-2\nu}{2} \end{bmatrix} \tag{4.13}$$

The following literature review shows some applications of linear elastic theory to the solution of soil mechanics problems.

In 1965, Clough and Rashid[17] investigated the stress distribution in a semi-infinite elastic half-space subjected to a concentrated point load. Brown and King[12] then developed a program for the approximate study of the stability of an embankment, and Geertsma[41] offered a model for the investigation of subsidence based on the poroelasticity theory. In

1969 Sandhu and Wilson [73] were the first to study linear elastic consolidation in a finite element context.

A procedure for determining values of E and ν from one-dimensional compression tests was then developed by Penman et al.[64] The values of the material properties obtained were used for the prediction of the displacement in a large embankment dam during construction.

Cole and Burand [21] used a trial-and-error technique by matching numerically the observed displacements of an excavation and a retaining wall to obtain the Young's modulus that was then used to predict stresses and displacements resulting from future excavations in the same soil. A similar technique was presented in 1974 by Gambolati et. al.[40] for the simulation of the subsidence of Venice.

These examples, which are far from exhaustive, demonstrate that linear elastic analyses can give reasonable solutions for a single load path if sufficient care is taken in determining the material properties. This is also true for the investigation of surface subsidence providing only compaction takes place.[80]

Obviously, situations involving failure, such as non-recoverable strains upon unloading, require either a variable elasticity approach or preferably an elastoplastic or visco-plastic relationship.

4.3 Elastoplastic models.

4.3.1 Constitutive law.

The theoretical approach relates stress and strain increments for a general elastoplastic material, allowing for strain hardening as a function of volumetric plastic strain. Both the Mohr-Coulomb and critical state yield surfaces will be considered in this section. The critical state component requires an associated flow rule while the Mohr-Coulomb component may also consider a non-associated flow rule. A more extensive development of the theory presented in this section can be found elsewhere, e.g. Nayak and Zienkiewicz, [58] Naylor [60] and Humpheson[45].

Yield criterion defines the limits of elasticity under any possible combination of stresses. When written in terms of stress components, this is called a yield function and represents a surface in the n - dimensional stress space, which separates the elastic state from an outer zone of impermissible stress states. The position and size of the surface is dependent upon the initial yield surface and the hardening law which specifies the manner in which the surfaces change during plastic flow. Hardening is considered here to be dependent only on plastic strain ε^p. The yield surface can therefore be defined as

$$F(\sigma', \varepsilon^p) = 0 \tag{4.14}$$

Irreversible (plastic) straining occurs when the stress state reaches the yield surface, while for $F < 0$ the behaviour is elastic.

The direction of the plastic strains is defined by the plastic potential surface $Q(\sigma')$. The normal to this surface, passing through the current stress point, provides the flow rule

$$d\varepsilon^p = d\lambda \frac{\partial Q}{\partial \sigma'} \tag{4.15}$$

where $d\lambda$ is a yet unknown scalar. If $F = Q$, the flow rule is said to be associated.

Differentiating equation (4.14) gives the relationship

$$\left\{\frac{\partial F}{\partial \sigma'}\right\}^T d\sigma' + \left\{\frac{\partial F}{\partial \varepsilon^p}\right\}^T d\varepsilon^p = 0 \tag{4.16}$$

If the flow rule (4.15) is taken into account, then equation (4.16) becomes

$$\left\{\frac{\partial F}{\partial \boldsymbol{\sigma}'}\right\}^T d\boldsymbol{\sigma}' + \left\{\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p}\right\}^T d\lambda \frac{\partial Q}{\partial \boldsymbol{\sigma}'} = 0 \tag{4.17}$$

Hence

$$d\lambda = -\frac{\left\{\frac{\partial F}{\partial \boldsymbol{\sigma}'}\right\}^T d\boldsymbol{\sigma}'}{\left\{\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p}\right\}^T \frac{\partial Q}{\partial \boldsymbol{\sigma}'}} \tag{4.18}$$

Once the elastic limit has been exceeded for an elastoplastic model, the total strain $d\varepsilon$ is expressed as the sum of the elastic and plastic components $d\{\varepsilon^e\}$ and $d\{\varepsilon^p\}$

$$d\boldsymbol{\varepsilon} = d\boldsymbol{\varepsilon}^e + d\boldsymbol{\varepsilon}^p = (D^e)^{-1} d\boldsymbol{\sigma}' + d\lambda \frac{\partial Q}{\partial \boldsymbol{\sigma}'} \tag{4.19}$$

where D^e is the tangential elastic stiffness matrix.

In the present context only linear elasticity is considered, so that D^e is defined by equation (4.13).

Pre-multiplying equation (4.19) by $\{\partial F/\partial \boldsymbol{\sigma}'\}^T D^e$ and substituting for $\{\partial F/\partial \boldsymbol{\sigma}'\}^T d\boldsymbol{\sigma}'$ by means of equation (4.17), leads to the following:

$$\left\{\frac{\partial F}{\partial \boldsymbol{\sigma}'}\right\}^T D^e d\boldsymbol{\varepsilon} = -\left\{\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p}\right\}^T \frac{\partial Q}{\partial \boldsymbol{\sigma}'} d\lambda + \left\{\frac{\partial F}{\partial \boldsymbol{\sigma}'}\right\}^T D^e \frac{\partial Q}{\partial \boldsymbol{\sigma}'} d\lambda \tag{4.20}$$

The term $d\lambda$ is now obtained from equation (4.20) and may be substituted into equation (4.19). Pre-multiplying by D^e and rearranging gives the equation defining the tangent elastoplastic modulus matrix D^{ep}

$$d\sigma' = \left[D' - \frac{D' \frac{\partial Q}{\partial \sigma'} \left\{ \frac{\partial F}{\partial \sigma'} \right\}^T D'}{-\left\{ \frac{\partial F}{\partial \varepsilon^p} \right\}^T \frac{\partial Q}{\partial \sigma'} + \left\{ \frac{\partial F}{\partial \sigma'} \right\}^T D' \frac{\partial Q}{\partial \sigma'}} \right] d\varepsilon \tag{4.21}$$

The construction of elastoplastic constitutive matrix requires evaluation of vectors $\partial Q/\partial \sigma$, $\partial F/\partial \sigma'$ and $\partial F/\partial \varepsilon^p$. These vectors will be formulated in sections 4.3.3 and 4.3.4 for both the Mohr-Coulomb and critical state yield surfaces.

The surfaces will be defined using stress invariants p, q, and θ which were introduced in section 4.2. It is therefore necessary to evaluate the derivatives of these invariants with respect to σ'.

From the definition of the invariants it follows that

$$\frac{\partial p}{\partial \sigma'} = -\begin{Bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.22a}$$

$$\frac{\partial p}{\partial \sigma'} = \frac{1}{2q} \begin{Bmatrix} 2\sigma_x - \sigma_y - \sigma_z \\ 2\sigma_y - \sigma_x - \sigma_z \\ 2\sigma_z - \sigma_y - \sigma_x \\ 6\tau_{xy} \\ 6\tau_{yz} \\ 6\tau_{xz} \end{Bmatrix} \tag{4.22b}$$

Differentiating equation (4.4) yields

$$3\cos\theta_0 d\theta_0 = -\frac{27}{2q^3}\left(dj_3 - \frac{3J_3}{q}dq\right) \tag{4.22c}$$

so that

$$\frac{\partial\theta}{\partial\boldsymbol{\sigma}'} = \frac{\partial\theta_0}{\partial\boldsymbol{\sigma}'} = \frac{9}{2q^3\cos 3\theta_0}\left[\frac{3J_3}{q}\frac{\partial q}{\partial\boldsymbol{\sigma}'} - \frac{\partial J_3}{\partial\boldsymbol{\sigma}'}\right] \tag{4.22d}$$

Only plane strain, plane stress and axial symmetry will be considered in the elastoplastic computer code used here. Therefore, σ_{yz} and $\sigma_{zx} = 0$ so that

$$|J_3| = (\sigma_z + p)\left\{(\sigma_x + p)(\sigma_y + p) - \tau_{xy}^2\right\} \tag{4.23a}$$

and

$$\frac{\partial|J_3|}{\partial\boldsymbol{\sigma}'} = \begin{Bmatrix} (\sigma_y + p)(\sigma_z + p) \\ (\sigma_x + p)(\sigma_z + p) \\ (\sigma_x + p)(\sigma_y + p) - \tau_{xy}^2 \\ -2(\sigma_z + p)\tau_{xy} \\ 0 \\ 0 \end{Bmatrix} + \frac{1}{9}\begin{Bmatrix} q^2 \\ q^2 \\ q^2 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.23b}$$

4.3.2 Solution procedures.

Equation (4.21) is now used with the tangential stiffness method. In consolidation analyses the tangential stiffness matrix $[K_T]$ varies as consolidation proceeds and the plastic regions develop. The method therefore requires the repeated formation and solution of the stiffness

matrix. Since the modification of $[K_T]$ at each time step can prove expensive, its value should be amended at selected time levels only. An alternative to the tangential stiffness method is the initial stress method described by Zienkiewicz et. al.[94] In this procedure the initial stiffness is assumed to be constant, and the excess stress by which the yield surface is exceeded is redistributed elastically by calculating equivalent nodal forces due to this stress. Since the initial stiffness matrix is used throughout, elastic unloading is automatically achieved. In the context of consolidation analyses this procedure was used by Siriwardance and Desai.[82]

4.3.3 Mohr-Coulomb yield surface.

Mohr's theory of failure involves the construction of an envelope to all possible circles of stress that can be drawn for a particular problem. These envelopes are generally curved but are usually replaced by a straight line. This is equivalent to assuming that the soil conforms to the Coulomb failure criterion which states that there is a linear relationship between the shear stress τ at failure and the normal stress, σ_n (see Figure 4.3):

$$\tau = c + \sigma_n \tan\phi \qquad (4.24)$$

where c = apparent cohesion

and

ϕ = angle of internal friction.

From Figure 4.3, it may be deduced that

$$\sin\phi = \frac{-\frac{1}{2}(\sigma_1 - \sigma_3)}{\frac{c}{\tan\phi} - \frac{(\sigma_1 + \sigma_3)}{2}} \qquad (4.25)$$

or, by rearranging,

$$-(\sigma_1 - \sigma_3) = 2c\cos\phi - (\sigma_1 + \sigma_3)\sin\phi \qquad (4.26)$$

where σ_1 and σ_3 are the major and minor principal stresses at failure.

This yield criterion is independent of the intermediate principal stress σ_2 and is therefore not completely representative of the true behaviour. This criticism applies to any yield criterion that can be represented by a single line in a two-dimensional stress space.

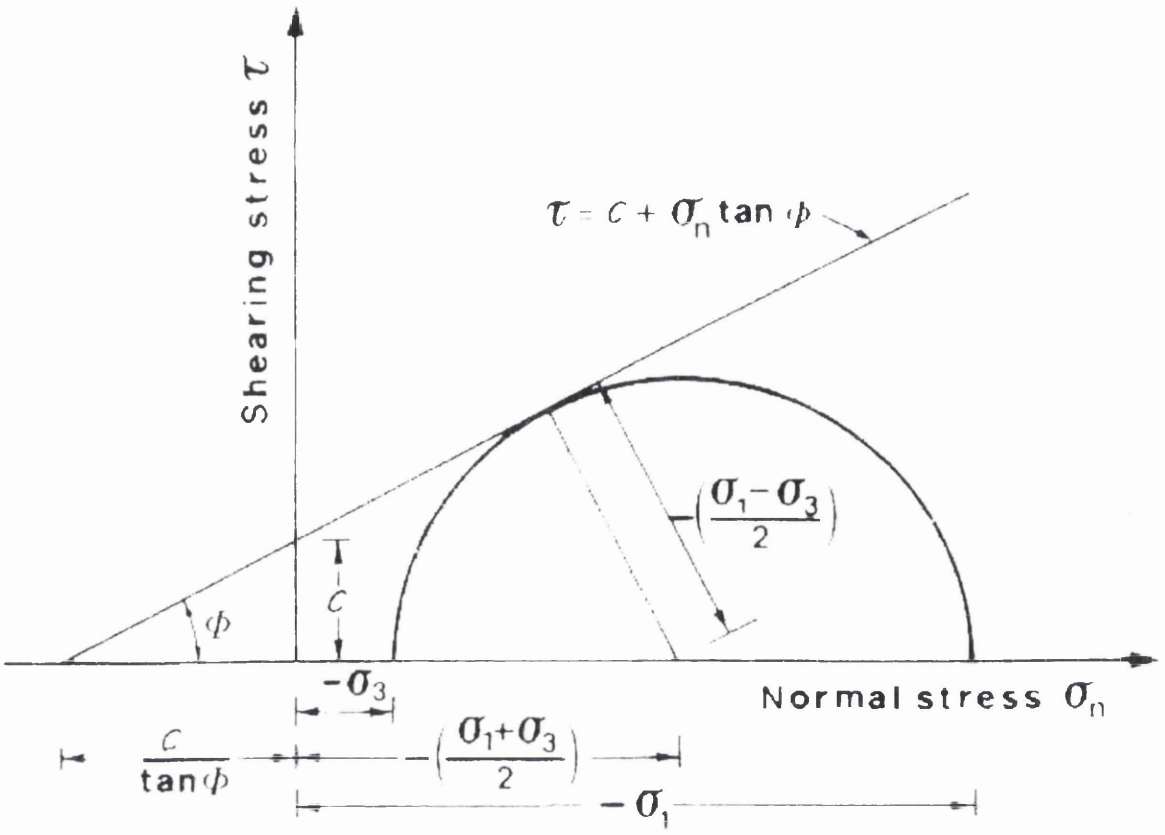


Figure 4.3 Mohr-Coulomb failure envelope

From equation (4.6) the sum and the difference of the maximum and minimum principal stresses are

$$\sigma_1 - \sigma_3 = \frac{2}{\sqrt{3}} q \cos\theta_0 \qquad (4.27a)$$

$$\sigma_1 + \sigma_3 = 2p + \frac{2}{3} q \sin\theta_0 \qquad (4.27b)$$

After substitution into the equation (4.26), the following expression for the Mohr-Coulomb failure surface becomes

$$F = \left(\sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\phi\right) q - 3p\sin\phi - 3c\cos\phi = 0 \qquad (4.28)$$

In a principal stress space, equation (4.28) represents an irregular right hexagonal pyramid, the axis of which lies along the space diagonal, as in Figure 4.4(a). The $\pi-$ plane section of the yield surface is shown in Figure 4.4(b).

The plastic potential surface, $Q(p,q,\theta_0)=0$, passing through the current stress point can be obtained in a similar manner. The direction of plastic straining in the τ,σ_n plane is usually expressed by the normal to a line defined as

$$\tau = \bar{c} + \sigma_n \tan\psi \qquad (4.29)$$

where the value of $\bar{c}$ ensures that the line passes through the current stress point, and ψ is the dilatancy angle. The potential surface Q, in terms of p, q, θ_0, can be obtained in the same way as the yield surface and results in the following expression:

$$Q = \left(\sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\psi\right) q - 3p\sin\psi - 3\bar{c}\cos\psi = 0 \qquad (4.30)$$

Equations (4.28) and (4.30) may be applied to either total or effective stresses by taking into account the appropriate material properties.

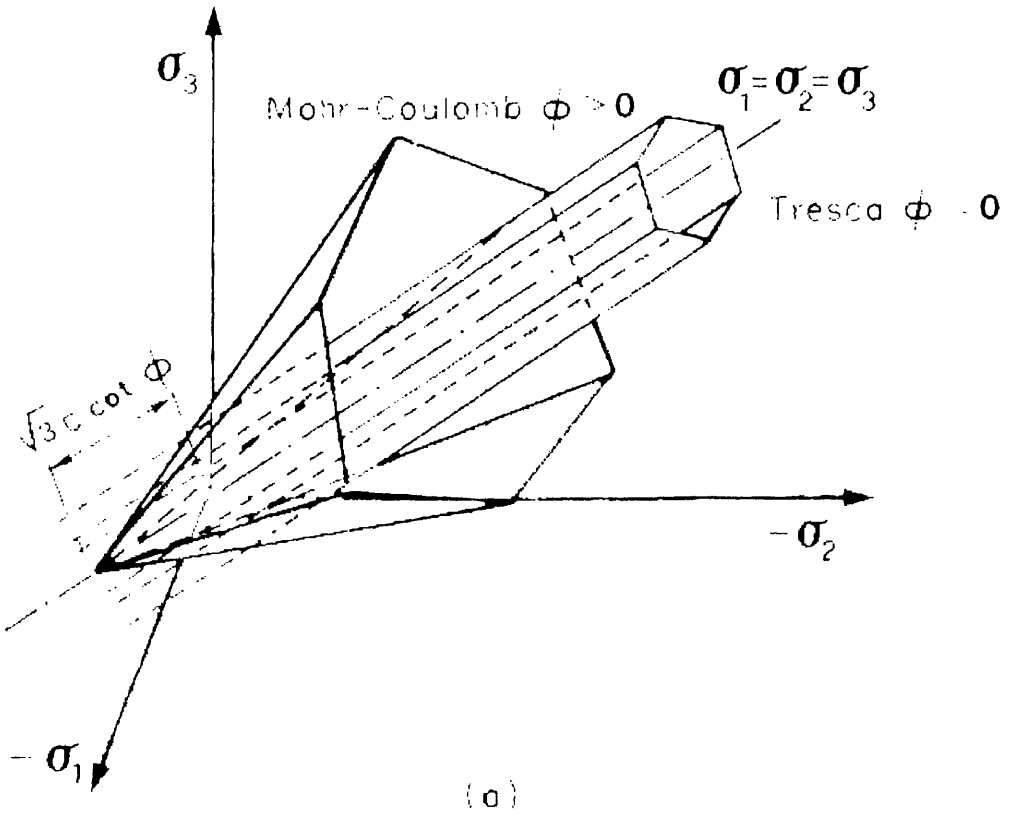
When $\phi=\psi=0$, which is the case in the total stress analysis of undrained saturated soil, both surfaces assume the form of a Tresca prism. In the

principal stress space this surface is represented by a regular hexagonal cylinder with its axis also lying along the space diagonal (see Figures 4.4(a) and (b)).

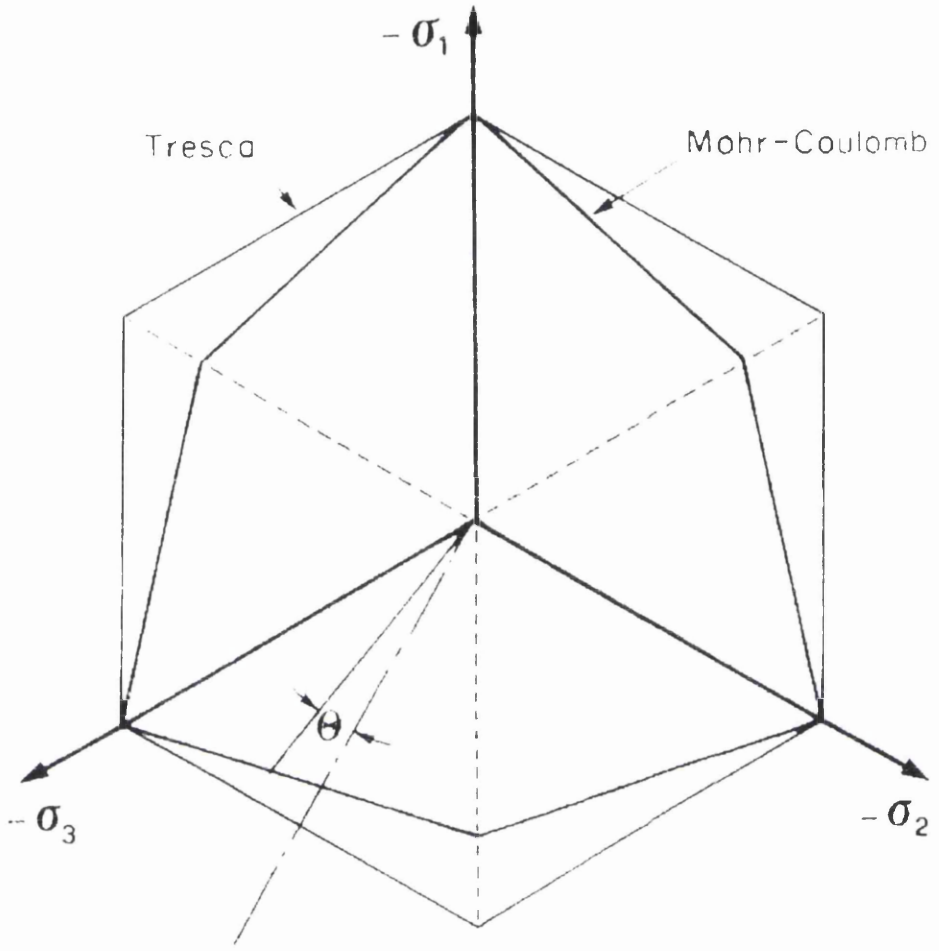
Strain hardening/softening may be dealt with by making c a function of volumetric plastic strain by using the equation

$$\frac{dc}{d\varepsilon_v^p} = \frac{c}{\chi} \tag{4.31}$$

where χ is a constant.



(a)



(b)

Figure 4.4 (a) Mohr-Coulomb and Tresca Yield surfaces in principal stress space;

(b) intersection with π-plane

The vectors $\partial F/\partial \boldsymbol{\sigma}'$, $\partial Q/\partial \boldsymbol{\sigma}'$ and $\partial F/\partial \boldsymbol{\varepsilon}^p$ required for the constitutive matrix of equation (4.21) can now be derived. The chain rule of differentiation gives

$$\frac{\partial F}{\partial \boldsymbol{\sigma}'} = \frac{\partial F}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}'} + \frac{\partial F}{\partial p}\frac{\partial q}{\partial \boldsymbol{\sigma}'} + \frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial \boldsymbol{\sigma}'} \tag{4.32}$$

The three derivatives $\partial F/\partial p$, $\partial F/\partial q$, $\partial F/\partial \theta$ are evaluated as follows:

$$\frac{\partial F}{\partial p} = -3\sin\phi \tag{4.33a}$$

$$\frac{\partial F}{\partial q} = \sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\phi \tag{4.33b}$$

$$\frac{\partial F}{\partial \theta} = -\left(\sqrt{3}\sin\theta_0 + \cos\theta_0 \sin\phi\right) q \tag{4.33c}$$

whilst $\partial p/\partial \boldsymbol{\sigma}$, $\partial q/\partial \boldsymbol{\sigma}$ and $\partial \theta/\partial \boldsymbol{\sigma}$ are given by equations (4.22)

Similarly,

$$\frac{\partial Q}{\partial \boldsymbol{\sigma}} = \frac{\partial Q}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}} + \frac{\partial Q}{\partial q}\frac{\partial q}{\partial \boldsymbol{\sigma}} + \frac{\partial Q}{\partial \theta}\frac{\partial \theta}{\partial \boldsymbol{\sigma}} \tag{4.34}$$

where $\partial Q/\partial p$, $\partial Q/\partial q$ and $\partial Q/\partial \theta$ are given by substituting ψ for ϕ in the above expressions for $\partial F/\partial p, \partial Q/\partial q$, and $\partial F/\partial \theta$.

Also, from equations (4.28) and (4.31)

$$\begin{aligned}\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p} &= \frac{\partial F}{\partial \varepsilon_v^p}\frac{\partial \varepsilon_v^p}{\partial \boldsymbol{\varepsilon}^p} + \left[\frac{\partial F}{\partial \varepsilon_q^p}\frac{\partial \varepsilon_q^p}{\partial \boldsymbol{\varepsilon}^p} = 0\right] \\ &= \frac{\partial F}{\partial c}\frac{\partial c}{\partial \varepsilon_v^p}\frac{\partial \varepsilon_v^p}{\partial \boldsymbol{\varepsilon}^p} = 3\cos\phi \frac{\partial c}{\partial \varepsilon_v^p}[-1,-1,-1,0,0,0]^T\end{aligned} \tag{4.35}$$

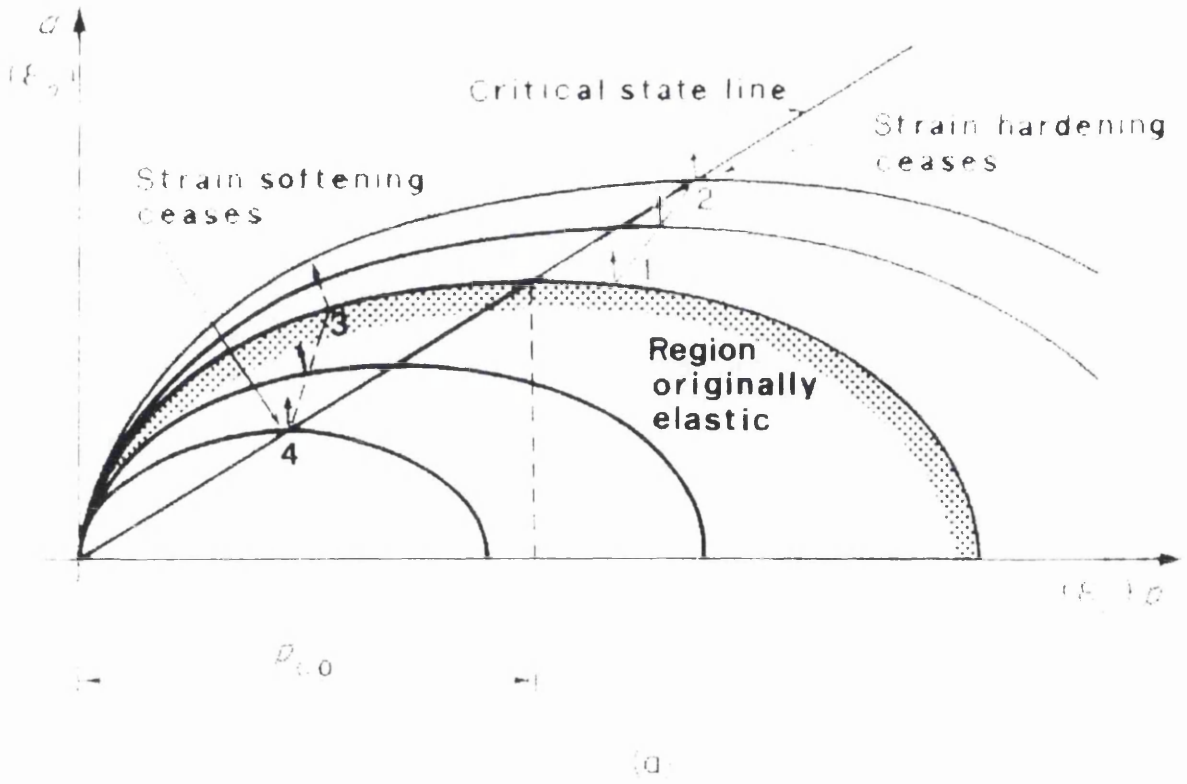
Since $[-1,-1,-1,0,0,0]\partial Q/\partial \boldsymbol{\sigma} = \partial Q/\partial p$, it follows that

$$\left\{\frac{\partial F}{\partial \varepsilon^p}\right\}^T \frac{\partial Q}{\partial \sigma'} = \frac{3c}{\chi}\cos\phi\frac{\partial Q}{\partial p} = -\frac{qc}{\chi}\cos\phi\sin\psi \qquad (4.36)$$

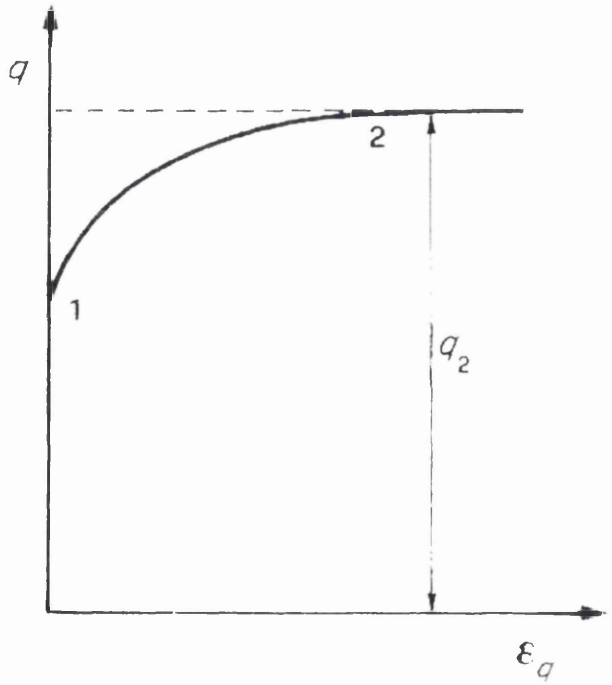
which is needed for the constitutive matrix.

4.3.4 Critical state model.

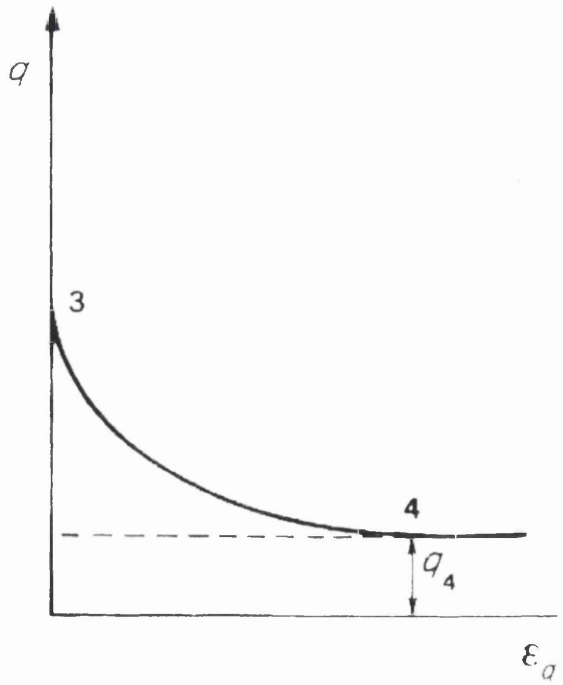
The critical state model is a form of elastoplastic isotropic strain hardening law. It introduces a distinction between yielding and ultimate collapse by using the concept of a critical state line in conjunction with a strain-dependent yield surface. A soil is at the critical state[69] if, during continuous deformation, there is no change in both the void ratio and effective stress components. In this model a soil undergoing shear deformation can pass through a yield point without collapse and continue to deform until eventually the critical state line is reached. Where ideal plasticity conditions exist, the soil continues to deform without further change of void ratio or stress. Starting from two alternative assumptions regarding the dissipation of energy during plastic yielding, Roscoe et al[70] and Schofield and Wroth[78] proposed the 'Cam clay' model in 1963. Roscoe and Burland[68] offered the 'modified Cam clay' model five years later.


Critical state line
Strain hardening ceases
Strain softening ceases
Region originally elastic
1
2
3
4

(a)


q
1
2
q_2
ε_q

(b)


q
3
4
q_4
ε_q

(c)

Figure 4.5 (a) Modified Cam clay model in the space of the two stress invariants p and q
(b) strain hardening behaviour
(c) strain softening behaviour

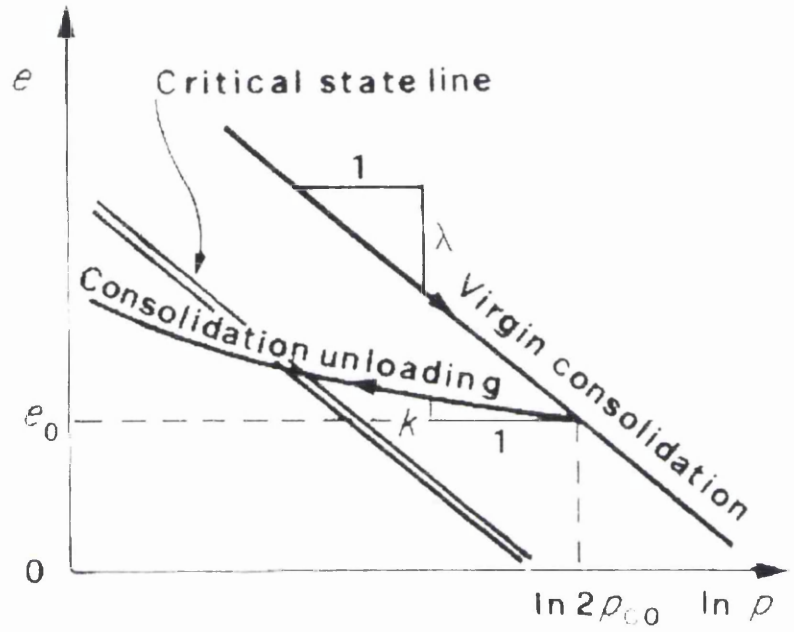


Figure 4.6 Consolidation parameters λ and k

4.3.4.1 Modified Cam clay model.

The modified Cam clay model fits experimental data quite satisfactorily. Its yield surface is an ellipse in the $p,q-plane$ shown in all of Figure 4.5, and is denoted by the equation

$$F=\frac{q^2}{M_{CS}^2}-2PP_c\left(\varepsilon_V^p\right)+p^2=0 \qquad (4.37)$$

where M_{cs} is the slope of the failure line in the $p,q-plane$, and $P_c(\varepsilon_V^p)$ is the current semi-diameter of the ellipse in the p-direction. The full

surface is a surface of revolution about the q-axis and is therefore defined by p and q only.

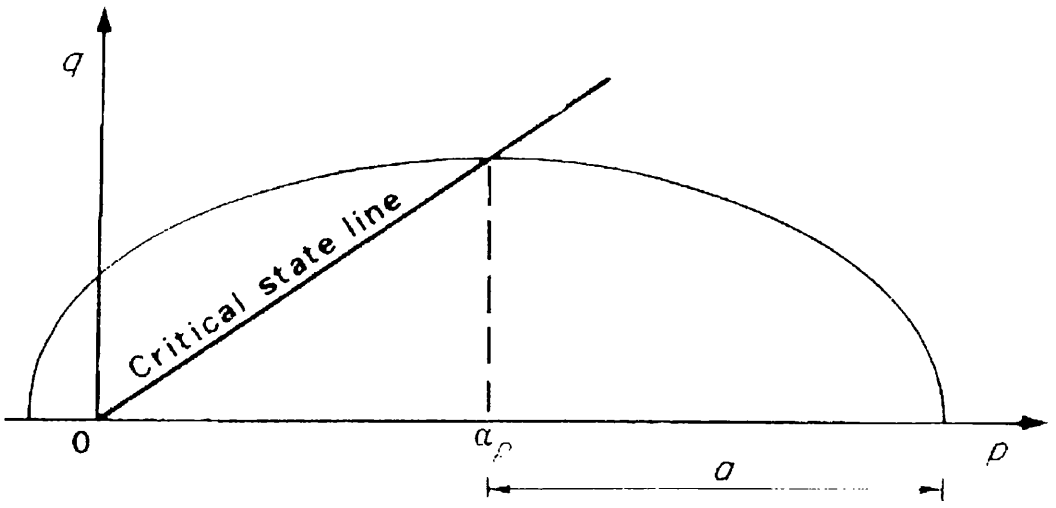


Figure 4.7 $p-q-\vartheta$ Critical state model

As the ellipse changes in magnitude, the locus of the critical state points is a pyramid with its apex at the origin shown by the critical state line in the $pq-plane$ as in Figure 4.5(a). The flow rule is associative, F=Q so the principle of normality applies to the yield surface. Since the surface is smooth, the direction of plastic straining is uniquely defined for every point of the surface. At the intersection of the critical state line and the ellipse the normal to the yield surface is vertical. At this point no component of plastic volumetric strain exists and all the plastic strain is distortional, hence the soil can deform at a constant volume.

The yield surface is therefore strain dependent and expands or contracts as the soil hardens or softens. Strain hardening is associated with compaction and strain softening with a volume increase. The initial size of the ellipse is governed by the maximum pre-consolidation pressure $2P_{co}$ to which the soil has previously been subjected. If the soil has been over-consolidated at some time in its history, then P_{co} may be quite large and the soil could sustain substantial loads before any yielding occurred. For a stress path of type 1-2 in Figure 4.5(b) the plastic strain vector

normal to the ellipse produces a plastic volumetric decrease which causes the soil to harden. The ellipse expands until eventually position 2 is reached, at which point no further volumetric strain occurs. The soil flows as a frictional fluid with constant volume.

The stress path 3-4 in Figure 4.5(c) shows a strain softening behaviour due to the expansion of the material. Consequently, the ellipse decreases in size and eventually, at point 4, the volume change limit is reached and collapse occurs at constant volume.

The strain hardening law uses the consolidation parameters λ and κ obtained by isotropic loading and unloading of normally consolidated soil. The parameter λ is the slope at the void ratio versus $\ln p$ plot during loading whilst κ is the initial value of the slope during rebound, as shown in Figure 4.6. From the geometry of Figure 4.6 it may be seen that the relationship between the plastic component of the void ratio change and the mean stress, p, is given by:

$$(e-e_0)^p = -(\lambda-k)(\ln 2p_c - \ln 2p_{c0})$$

or

$$\ln(p_c / p_{c0}) = -\frac{(e-e_0)^p}{(\lambda-k)} \qquad (4.38a)$$

The changes in volumetric strain are related to changes in void ratio by means of the following equation:

$$\varepsilon_v - \varepsilon_{v0} = -\frac{e-e_0}{1+e_0} \qquad (4.38b)$$

Hence, equation (4.38a) may be rewritten in the form

$$p_c = p_{co} \exp\left(\varepsilon_v - \varepsilon_{v0}\right)^p / \chi \tag{4.38c}$$

where $\chi = \lambda - \kappa / 1 - e_0$, is an empirical constant.

If the accumulated plastic volumetric strain is denoted h as follows

$$h = \int d\varepsilon_v^p = (\varepsilon_v - \varepsilon_{v0})^p \tag{4.38d}$$

the hardening law assumes the form

$$p_c = p_{c0} \exp \frac{h}{\chi} \tag{4.38e}$$

The derivatives $\partial F / \partial \boldsymbol{\sigma}'$ and $\partial F / \partial \boldsymbol{\sigma}^p$ required to set up the constitutive matrix are next obtained.

From equation (4.37) it follows that

$$\frac{\partial F}{\partial \boldsymbol{\sigma}'} = \frac{\partial F}{\partial p} \frac{\partial p}{\partial \boldsymbol{\sigma}'} + \frac{\partial F}{\partial q} \frac{\partial q}{\partial \boldsymbol{\sigma}'} \tag{4.39a}$$

where

$$\frac{\partial F}{\partial p} = 2\left(p - p_c\right) \tag{4.39b}$$

$$\frac{\partial F}{\partial q} = \frac{2q}{M_{CS}^2} \tag{4.39c}$$

and

$\frac{\partial p}{\partial \boldsymbol{\sigma}}$ and $\frac{\partial q}{\partial \boldsymbol{\sigma}}$ are given by equations (4.22).

From equation (4.37), it follows that

$$\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p} = \frac{\partial F}{\partial \varepsilon_v^p}\frac{\partial \varepsilon_v^p}{\partial \boldsymbol{\varepsilon}^p} + \left[\frac{\partial F}{\partial \varepsilon_q^p}\frac{\partial \varepsilon_q^p}{\partial \boldsymbol{\varepsilon}^p} = 0\right]$$

$$= \left(-2p\frac{\partial p_c}{\partial \varepsilon_v^p}\right)\begin{Bmatrix}-1\\-1\\-1\\0\\0\\0\end{Bmatrix} \qquad (4.40)$$

From equation (4.38c), the following may be obtained:

$$\frac{dp_c}{p_c} = \frac{d\varepsilon_v^p}{\chi} \qquad (4.41)$$

Hence, equation (4.40) becomes

$$\frac{\partial F}{\partial \boldsymbol{\varepsilon}^p} = \left(\frac{-2pp_c}{\chi}\right)\begin{Bmatrix}-1\\-1\\-1\\0\\0\\0\end{Bmatrix} \qquad (4.42)$$

From equations (4.21), (4.37) and (4.38c) it can be seen that before defining the parameters of the modified Cam clay model, knowledge of $E, \nu, p_{c0}, M_{cs}, e_0, \lambda$ and κ is required. The next section will discuss a

generalized critical state model involving the third stress invariant which requires one more parameter.

4.3.4.2 $p-q-\theta$ Critical state model

The generalization of the critical state model to include the effect of the variation of the third stress invariant θ is achieved by letting the slope of the critical state line vary with θ. In 1975, Zienkiewicz et al [93] developed an elliptical model whose π-plane section was the same as that of the Mohr-Coulomb surface. An alternative model proposed by Humpheson [45] uses the critical state ellipse only in the sub-critical region while the super-critical region may be cut off by a Mohr-Coulomb surface.

The model dealt with here is used by Norris [63] and consists of the full ellipse with a Mohr-Coulomb, π-plane section and with the critical state line passing through the origin is defined as

$$F=\left[p-\alpha_p\left(\varepsilon_v^p\right)\right]^2+\left[q/n(\theta)\right]^2-a^2\left(\varepsilon_v^p\right)=0 \qquad (4.43)$$

where $\alpha_p\left(\varepsilon_v^p\right)$ is the p-coordinate of the centre of the elliptical surface

$n(\theta)$ is the ratio of the diameters in the q and p=directions, defined by the Mohr-Coulomb equation for c=0

$a\left(\varepsilon_v^p\right)$ is the semi-diameter of the ellipse in the p=direction

(see Figure 4.7)

Once again, plastic yielding is the associated form and strain hardening depends on the plastic changes of void ratio or volumetric strain. From inspection of the critical state line passing through the origin, it follows

that

$$\frac{d\alpha_p}{\alpha_p} = \frac{da}{a} \tag{4.44}$$

The hardening rule can be obtained in a similar manner[63] as shown in the previous section, resulting in

$$\alpha_p = \alpha_{p0} \exp\frac{h}{\chi} \tag{4.45a}$$

and

$$a = a_0 \exp\frac{h}{\chi} \tag{4.45b}$$

Hence,

$$\frac{d\alpha_p}{\alpha_p} = \frac{da}{a} = \frac{d\varepsilon_v^p}{\chi} \tag{4.45c}$$

A formula for one more term, n, is required for the definition of the yield surface of equation (4.43).

It is assumed that the 'critical state' lies on a Mohr-Coulomb surface for which c equals zero. From equation (4.28), with c=0, it follows that

$$M_{cs} = \frac{\partial q}{\partial p} = \frac{3\sin\phi}{\sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\phi} \tag{4.46}$$

From the geometry of Figure 4.7 it may be seen that

$$n(\theta) = \frac{\alpha_p M_{cs}}{a} = \frac{\alpha_p}{a} \frac{3\sin\phi}{\sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\phi} \tag{4.47}$$

The derivatives $\partial F / \partial \boldsymbol{\sigma}'$ and $\partial F / \partial \boldsymbol{\varepsilon}^p$ required for the constitutive matrix are now derived. From equations (4.43) (4.47) it can be seen that

$$\frac{\partial F}{\partial \boldsymbol{\sigma}'} = \frac{\partial F}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}'} + \frac{\partial F}{\partial q}\frac{\partial q}{\partial \boldsymbol{\sigma}'} + \frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial \boldsymbol{\sigma}'} \tag{4.48a}$$

where

$$\frac{\partial F}{\partial p} = 2\left(p - \alpha_p\right) \tag{4.48b}$$

$$\frac{\partial F}{\partial q} = \frac{2q}{n^2} \tag{4.48c}$$

$$\frac{\partial F}{\partial \theta} = -\frac{2q^2}{n^3}\frac{dn}{d\theta} = -\frac{2q^2}{n^2}\frac{\sqrt{3}\sin\theta_0 + \cos\theta_0 \sin\phi}{\sqrt{3}\cos\theta_0 - \sin\theta_0 \sin\phi} \tag{4.48d}$$

and $\partial p / \partial \boldsymbol{\sigma}', q / \partial \boldsymbol{\sigma}'$ and $\partial \theta / \partial \boldsymbol{\sigma}'$ are given by equations (4.22).

From equations (4.43) and (4.45c) (see also equation (4.40))

$$\frac{\partial F}{\partial \boldsymbol{\mathcal{E}}^p}=\left[-2\left(p-\alpha_p\right)\frac{d\alpha_p}{d\varepsilon_v^p}-2a\frac{da}{d\varepsilon_v^p}\right]\begin{Bmatrix}-1\\-1\\-1\\0\\0\\0\end{Bmatrix}=-\frac{2}{\chi}\left[\left(p-\alpha_p\right)\alpha_p+a^2\right]\begin{Bmatrix}-1\\-1\\-1\\0\\0\\0\end{Bmatrix} \tag{4.49}$$

Finally, the product $\left\{\partial F/\partial \boldsymbol{\mathcal{E}}^p\right\}^T \partial Q/\partial \boldsymbol{\sigma}'$ appearing in the constitutive matrix is given by

$$\begin{aligned}\left\{\frac{\partial F}{\partial \boldsymbol{\mathcal{E}}^p}\right\}^T\frac{\partial Q}{\partial \boldsymbol{\sigma}'}&=-\frac{2}{\chi}\left[\left(p-\alpha_p\right)\alpha_p+a^2\right]\left(-\frac{\partial Q}{\partial \sigma_{11}'}-\frac{\partial Q}{\partial \sigma_{22}'}-\frac{\partial Q}{\partial \sigma_{33}'}\right)\\&=-\frac{2}{\chi}\left[\left(p-\alpha_p\right)\alpha_p+a^2\right]\frac{\partial Q}{\partial p}\end{aligned} \tag{4.50}$$

For associated flow, $\partial Q/\partial p=\partial F/\partial p=2\left(p-\alpha_p\right)$(equation (4.48b)) so that

$$\left\{\frac{\partial F}{\partial \boldsymbol{\mathcal{E}}^p}\right\}^T\frac{\partial Q}{\partial \boldsymbol{\sigma}'}=-\frac{4}{\chi}\left[\left(p-\alpha_p\right)\alpha_p+a^2\right]\left(p-\alpha_p\right) \tag{4.51}$$

4.4 Corners of yield and potential surfaces.

The Mohr-Coulomb potential surface and the elliptic $p-q-\theta$ surface show corners for $\theta_0=\pm\pi/6$, where the plastic strain direction is no longer uniquely defined. In the Mohr-Coulomb case there is a further

complexity when $q=0$ at the apex of the potential surface. In the case of associated plastic flow, Drucker[34] concluded that the plastic strain direction is confined by the normal to the conjoint potential surfaces. For simplicity, in the models described here the plastic strain direction is assumed to lie in the planes for which $\theta_0 = \pm\pi/6$ and at the apex of the Mohr-Coulomb potential surface, in the negative direction of the p-axis.[63]

4.5 Variation of permeability.

Until now, only the non-linear relationship between relative permeability and saturation has been taken into account. However, there is experimental evidence that even in saturated soil, with only one fluid phase flowing, the permeability is not constant.

In groundwater flow literature, the product

$$K = \frac{k\gamma}{\mu} \tag{4.52}$$

is usually called permeability, where μ is the dynamic viscosity of water and γ is the specific weight of water.

De Wiest[28] has shown that γ is a function of the pressure

$$\gamma = \gamma_0 \exp\left(\frac{p_0 + p}{K_w}\right) \tag{4.53}$$

where K_w is the bulk modulus of water.

The effect of the dependence of the permeability K on the pressure, via the specific weight of water, was investigated by Gambolati[36,37] in the case of one-dimensional vertical flow of groundwater. It was found that the effects on the pressure field are negligible unless very thick formations ($>10^4$m) and very high boundary pressures ($>5\times10^5$MPa) are considered. Such values are extremely unrealistic and therefore this source of non-linearity is disregarded.

Researchers have investigated other forms of non-linear permeability in their numerical models. For example, Finol and Farouq Ali[35] considered the case of a black oil system where water was immobile. They determined the variation of porosity and permeability with pressure in accordance with the following relationship:

$$\phi^{m+1} = \phi^m \left[1 + c_p \left(p_0^{m+1} - p_0^m\right)\right] \tag{4.54a}$$

$$k^{m+1} = k^m \left[1 + c_p \left(p_0^{m+1} - p_0^m\right)\right] \tag{4.54b}$$

where

$$c_p = \frac{c_m + (1-\phi)c_r}{\phi} \tag{4.54c}$$

c_m is the uniaxial compaction coefficient

c_r is the rock matrix compressibility

c_k is coefficient of permeability reduction

p_o is the pressure in the oil phase.

The uniaxial compaction coefficient c_m is defined as the formation compaction per unit change in pore pressure reduction and can be obtained from laboratory compressibility data by

$$c_m = \frac{1}{3}\frac{1+\nu}{1-\nu}(1-\beta)c_b \tag{4.55}$$

where

ν is Poisson's ratio

β is the ratio of rock matrix and rock bulk compressibility

c_b is the bulk compressibility.

Because of the lack of appropriate experimental data, c_k was arbitrarily taken to be equal to the uniaxial compaction coefficient. In the case of one-dimensional consolidation, Monte and Kritz[57] found experimentally that a bilinear relationship between void ratio e and logarithm of permeability K represents soil behaviour fairly well. Such a relationship is shown in Figure 4.8.

If K_c and e_c are the critical values of permeability and void ratio at which the slope changes and if the initial void ratio is e_0 then for a void ratio e the permeability K is given by

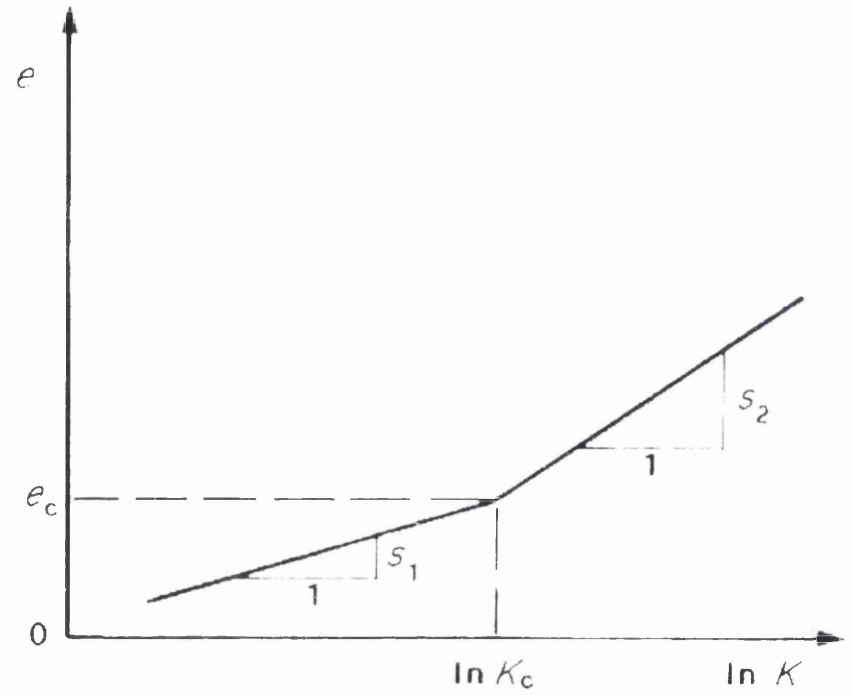


Figure 4.8 Relationship between void ratio and the logarithm of the permeability

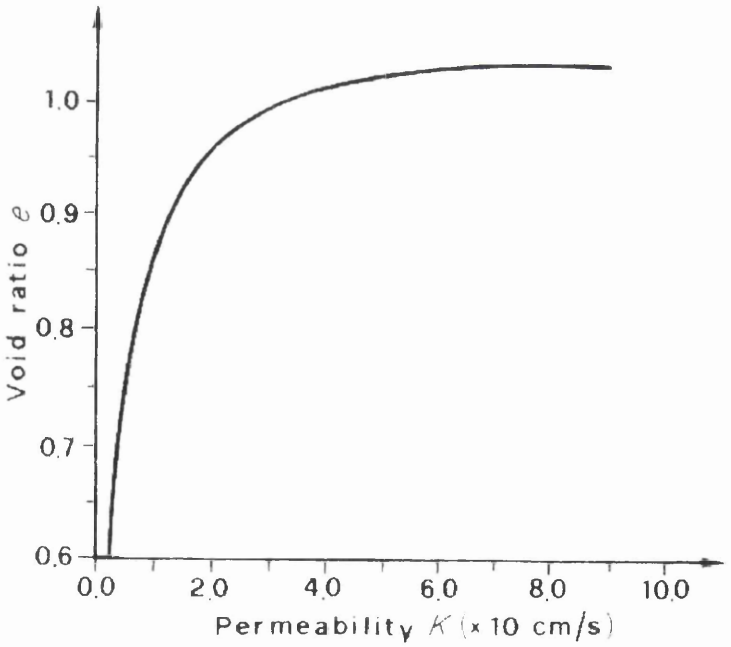


Figure 4.9 Relationship between void ratio and permeability, obtained experimentally

$$\ln\left(\frac{K}{K_c}\right)=\frac{e-e_c}{S} \tag{4.56}$$

where

$S = S_1, e < e_c$
$S = S_2, e > e_c$

Since the variation of K is less certain for more than one dimension, this procedure was used numerically by Norris [63] for the vertical direction only, which is assumed to be the direction of maximum strain.

The procedure utilized in this thesis was proposed in 1976 by Lewis et al [55] and involves an experimental permeability/void ratio relationship that is determined in the laboratory. An example of such a relationship is shown in Figure 4.9.

This relationship is represented by as many pairs of data points as required and intermediate values are obtained by interpolation. The model is fully defined by the settlement/void ratio relationship

$$\delta = d\frac{e_1 - e_2}{1+e_1} \tag{4.57}$$

where δ is the average settlement of each element

d is the initial depth

e_1 is the initial void ratio

e_2 is the final void ratio.

Both relationships apply to one-dimensional consolidation. Their validity to model the variation of the permeability K in more than one dimension is less certain and requires further research. The variable permeability

scheme is of practical use only if the expected consolidation is important. In the case of subsidence analysis the permeability variations appear to be negligible.

4.6 Concluding remarks.

Both linear and non-linear elastic and elastoplastic constitutive relationships, which are suitable for application in consolidation analysis and modelling of surface subsidence, have been outlined in this chapter. Variable permeability schemes have also been discussed in the last section. The stage is now set for the application of the procedures outlined in Chapters 2 and 3, together with the constitutive relationships.

Chapter 5

Numerical solution for the continuum theory of consolidation

Numerical solution of the continuum theory of consolidation is derived from the general formulations of the coupled governing equations (3.38) and (3.53)-(3.55) or (3.70)-(3.71) and (3.79). The mathematical procedures may be divided into two types: the uncoupled and the coupled solution. Numerical method of consolidation problem is needed for the coupled solution.[31,38,39,40,54]

5.1 Coupled solution for saturated one-phase flow in a deforming porous medium.

The particular form of the continuity equation (3.70), together with the equilibrium equation (3.38), is derived from the governing equations for soil mechanics problems. These equations have been shown to arise from Biot's[8,9] self-consistent theory and are presented here again for the sake of completeness. Since only one fluid phase is present, the terms of the subscript w have been omitted:

$$\int_{\Omega}\delta^{T}D_{T}\frac{\partial\varepsilon}{\partial t}d\Omega-\int_{\Omega}\delta\varepsilon^{T}m\frac{\partial\varepsilon}{\partial t}d\Omega+\int_{\Omega}\delta\varepsilon^{T}D_{T}m\frac{\partial\varepsilon}{\partial t}\frac{1}{3K_{S}}d\Omega$$
$$-\int_{\Omega}\delta\varepsilon^{T}D_{T}cd\Omega-\int_{\Omega}\delta\varepsilon^{T}D_{T}\frac{\partial\varepsilon}{\partial t}d\Omega-\frac{d\widehat{f}}{dt}=0 \qquad (5.1)$$

$$\bar{A} = -\nabla^T \{\frac{k}{\mu}\nabla(p+\rho gh) + (m^T - \frac{m^T D_T}{3K_S})\frac{\partial \varepsilon}{\partial t} + \frac{m^T D_T c}{3K_S}$$
$$+[\frac{1-\phi}{K_S} + \frac{\phi}{K_w} - \frac{1}{(3K_S)^2} m^T D_T m]\frac{\partial p}{\partial t} = 0 \qquad (5.2)$$

5.2 Boundary values of the solution.

A boundary value problem requires that the governing equations are satisfied within all points of a continuum (domain Ω) and that the boundary conditions are satisfied on the boundary Γ of the domain.

In equilibrium equations (5.1) the boundary conditions are already incorporated. Attention is therefore focused on the continuity equation. In this case, the boundary conditions satisfy

(a) the continuity of flow across the boundary

$$\bar{B} \equiv -n^T \frac{k}{\mu}\nabla(p+\rho gh) - q = 0 \qquad (5.3)$$

where n is the unit normal vector and q is the outflow rate per unit area of the boundary surface; and

(b) prescribed pore pressures

$$p = p^b \qquad (5.4)$$

To satisfy the condition that the continuity equation (5.2) apply throughout the continuum, and that equation (5.3) apply on the boundary requires that

$$\int_\Omega a^T \bar{A}\, d\Omega + \int_\Gamma b^T \bar{B}\, d\Gamma = 0 \qquad (5.5)$$

where a and b are a set of arbitrary functions since $\bar{A}$ and $\bar{B}$ are identically satisfied throughout their respective domains. Conversely, if equation (5.5) is valid for any arbitrary values of a and b, then the differential equations (5.2) and (5.3) must be satisfied at all points within and on the boundary of the continuum.[95]

The next step is to determine the variation of conditions in the continuum which will satisfy equations (5.1) and (5.5). Only for the very simplest of problems can this be achieved by exact analysis. For most practical problems some of form of approximation is needed. This is achieved by means of the finite element method as outlined in the next section.

5.3 Application of the finite element method.

The finite element method will be applied to equations (5.1) and (5.5) in terms of displacements and pore pressures. In equation (5.2) the appearance of second derivatives for $(p+\rho gh)$ necessitates a smooth distribution in space due to the integration of these variables. In order to overcome this limitation, a weak form of equation (5.2) is obtained by means of Green's theorem.

$$\int_{\Omega}\phi\frac{\partial\psi}{\partial x}d\Omega=-\int_{\Omega}\frac{\partial\phi}{\partial x}\psi d\Omega+\int_{\Gamma}\phi\psi n_x d\Gamma \tag{5.6}$$

where n_x is the direction cosine between the outward normal and the $x-$direction.

Upon substitution of equation (5.2) and equation (5.3), equation (5.5) becomes

$$\int_{\Omega}\{(\nabla a)^T\frac{k}{\mu}\nabla(p+\rho gh)+a^T[(m^T-\frac{m^TD_Tc}{3K_S}$$
$$+(\frac{1-\phi}{K_S}+\frac{\phi}{K_w}-\frac{1}{(3K_S)^2}m^TD_Tm)\frac{\partial p}{\partial t}]\}d\Omega \qquad (5.7)$$
$$-\int_{\Gamma}\{a^Tn^T\frac{k}{\mu}\nabla(p+\rho gh)+b^Tn^T\frac{k}{\mu}\nabla(p+\rho gh)+b^Tq\}d\Gamma=0$$

Since the values of a and b are arbitrary, we can make

$$b=-a$$

and thus eliminate some of the terms of the boundary integrals. Equation (5.7) therefore reduces to

$$\int_{\Omega}\{(\nabla a)^T\frac{k}{\mu}\nabla(p+\rho gh)+a^T[(m^T-\frac{m^TD_T}{3K_S})\frac{\partial\varepsilon}{\partial t}+\frac{m^TD_Tc}{3K_S}$$
$$+(\frac{1-\phi}{K_S}+\frac{\phi}{K_w}-\frac{1}{(3K_S)^2}m^TD_Tm)\frac{\partial p}{\partial t}]\}d\Omega+\int_{\Gamma}a^Tqd\Gamma=0 \qquad (5.8)$$

The finite element approximation is now applied to equations (5.1) and (5.8). The displacements and pore pressure are expressed in terms of their values $\bar{u}$ and $\bar{p}$ at a finite number of points in space. This procedure involves the division of the continuum into elements, and the expression of u and p within an element in terms of their values at a finite number of points within or on the boundary of that element. In order to ensure continuity of displacements and pore pressures between elements, it is necessary to place a sufficient number of nodes on the element boundary to satisfy the shape function $[N]$ being used. The expressions for u, ε and p take the form

$$u=N\bar{u} \qquad (5.9a)$$

$$\varepsilon = B\bar{u} \tag{5.9b}$$

$$p = \bar{N}\bar{p} \tag{5.9c}$$

The numbers and locations of the nodes are not necessarily the same for u and p.

The finite element discretization gives the result

$$\begin{aligned}
&\delta u^T \{ \int_\Omega B^T D_T B d\Omega \frac{d\bar{u}}{dt} - \int_\Omega B^T m \bar{N} d\Omega \frac{d\bar{p}}{dt} + \int_\Omega B^T D_T \frac{m}{3K_S} N d\Omega \frac{d\bar{p}}{dt} \\
&- \int_\Omega B^T D_T c d\Omega - \frac{1}{dt} \int_\Omega B^T D_T d\varepsilon_0 d\Omega \} \\
&- \delta u^T \{ \int_\Omega N^T \frac{db}{dt} d\Omega + \int_\Gamma N^T \frac{d\hat{t}}{dt} d\Gamma \} = 0
\end{aligned} \tag{5.10}$$

$$\begin{aligned}
&\int_\Omega (\nabla a)^T \frac{k}{\mu} \nabla \bar{N} d\Omega \bar{p} + \int_\Omega a^T (m^T - \frac{m^T D_T}{3K_S}) B d\Omega \frac{d\bar{p}}{dt} + \int_\Omega a^T \frac{m^T D_T c}{3K_S} \\
&+ \int_\Omega a^T (\frac{1-\phi}{K_S} + \frac{\phi}{K_W} - \frac{1}{(3K_S)^2} m^T D_T m) \bar{N} d\Omega \frac{d\bar{p}}{dt} + \int_\Gamma a^T q d\Gamma \\
&+ \int_\Omega a^T (\frac{1-\phi}{K_S} + \frac{\phi}{K_W} - \frac{1}{(3K_S)^2} m^T D_T m) \bar{N} d\Omega \frac{d\bar{p}}{dt} + \int_\Gamma a^T q d\Gamma \\
&+ \int_\Omega a^T \nabla^T \frac{k}{u} \nabla \rho g h d\Omega = 0
\end{aligned} \tag{5.11}$$

Equation (5.10) is valid for any value of the virtual displacement $\delta\{u\}$ and can be written as

$$K \frac{d\bar{u}}{dt} + L \frac{d\bar{p}}{dt} - C - \frac{d\bar{f}}{dt} = 0 \tag{5.12}$$

where

$$K = -\int_{\Omega} B^T D_T B d\Omega \tag{5.13a}$$

$$L = \int_{\Omega} B^T m \bar{N} d\Omega - \int_{\Omega} B^T D_T \frac{m}{3K_s} \bar{N} d\Omega \tag{5.13b}$$

$$C = -\int_{\Omega} B^T D_T c d\Omega \tag{5.13c}$$

$$df = -\int_{\Omega} N^T db d\Omega - \int_{\Gamma} N^T d\hat{t} d\Gamma - \int_{\Omega} B^T D_T d\varepsilon_0 d\Omega \tag{5.13d}$$

The form of the function a in equation (5.11) is still quite arbitrary and must be specified before equation (5.11) can be solved. It is desirable to choose a form which will increase the accuracy of the approximation used. For this purpose, the weighted residual procedure has been applied, using the Galerkin method.[92]

The function a is replaced by a finite number of functions within each element, which in the Galerkin method are identical to the shape function $\bar{N}$. In the solution of the present problem, and in many others, this method has the particular advantage of giving rise to symmetric matrices. Equation (5.11) now becomes

$$H\bar{p} + S\frac{d\bar{p}}{dt} + L^T \frac{d\bar{u}}{dt} - \bar{f} = 0 \tag{5.14}$$

where

$$H = \int_{\Omega} (\nabla \bar{N})^T \frac{k}{\mu} \nabla \bar{N} d\Omega \tag{5.15a}$$

$$S = \int_{\Omega} \overline{N}^T s \overline{N} d\Omega$$

with $$s = \frac{1-\phi}{K_S} + \frac{\phi}{K_w} - \frac{1}{(3K_S)^2} m^T D_T m \qquad (5.15b)$$

$$L^T = \int_{\Omega} N^T (m^T - \frac{m^T D_T}{3K_s}) B d\Omega \qquad (5.15c)$$

and

$$\overline{f} = -\int_{\Gamma} \overline{N}^T q d\Gamma - \int_{\Omega} \frac{\overline{N}^T}{3K_S} m^T D_T c d\Omega - \int_{\Omega} (\nabla \overline{N})^T \frac{k}{\mu} \nabla \rho g h d\Omega \qquad (5.15d)$$

It can be easily verified that the complete set of equations is symmetric if $[D_T]$ is symmetric.

The integration of these equations usually requires the use of numerical techniques. A standard method is that of Gaussian quadrature,[95] where the integrands are evaluated at specific points of the element and boundary surfaces and then weighted and summed. The procedure is carried out in terms of a set of local coordinates ξ, η and ζ having values of ± 1 on the element boundaries.

The global coordinates $\{x\}$ are expressed in terms of the nodal coordinates $\{X\}$ by a relationship of the form

$$x = WX$$

where W is a set of weighting functions in terms of the local coordinates ξ,η and ζ.

The shape functions $\{N\}$, and $\{\overline{N}\}$ are also expressed as functions of the local coordinates, and if they are identical with W, then the net result is the so-called isoparametric element family.[92] The Cartesian derivatives of the shape function, $d\Omega$, and $d\Gamma$ are also expressed in terms of local coordinates.

Since the discretization in space has been carried out, equations (5.12) and (5.14) now represent a set of ordinary differential equations in time. For convenience, the equations are written in the following form:

$$\begin{bmatrix} 0 & 0 \\ 0 & H \end{bmatrix} \begin{Bmatrix} \overline{u} \\ \overline{p} \end{Bmatrix} + \begin{bmatrix} K & L \\ L^T & S \end{bmatrix} \frac{d}{dt} \begin{Bmatrix} \overline{u} \\ \overline{p} \end{Bmatrix} = \begin{Bmatrix} \frac{df}{dt} + C \\ \overline{f} \end{Bmatrix} \tag{5.16}$$

The values of $\{\overline{u}\}$ and $\{\overline{p}\}$ at different values in time may now be obtained by means of appropriate time-stepping algorithms.

5.4 Discretization in time

The method used for time discretization may be regarded as a one-dimensional finite element scheme, as distinct from spatial discretization (Kantorovich type approach).[63,92] The time domain is divided into a number of elements or steps, and integration is carried out for each step to obtain its changes of the parameters $\{\overline{u}\}$ and $\{\overline{p}\}$. The step-by-step integrations may then be summed to determine the total change of the parameters. Integration takes the same form as used for the spatial

integration, i.e., if $F=0$ then $\int \bar{g} F dt = 0$, where $\bar{g}$ is an arbitrary function of time. When applied to equation (5.16), this method requires the solution of the equations

$$\int_k^{k+\Delta t_k} \bar{g} \begin{bmatrix} 0 & 0 \\ 0 & H \end{bmatrix} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix} dt + \int_k^{k+\Delta t_k} \bar{g} \begin{bmatrix} K & L \\ L^T & S \end{bmatrix} \frac{d}{dt} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix} dt = \int_k^{k+\Delta t_k} \bar{g} \begin{Bmatrix} \frac{df}{dt} + C \\ \bar{f} \end{Bmatrix} dt \tag{5.17}$$

Where Δt_k is the length of the k^{th} time step.

The first-order time derivatives of $\{\bar{u}\}$ and $\{\bar{p}\}$ may be approximated by assuming a linear variation of $\{\bar{u}\}$ and $\{\bar{p}\}$ within each step, as follows:

$$\begin{bmatrix} \bar{u} & \bar{p} \end{bmatrix} = \begin{bmatrix} N_1^t & N_2^t \end{bmatrix} \begin{bmatrix} \bar{u}^t & \bar{p}^t \\ \bar{u}^{t+\Delta t} & \bar{p}^{t+\Delta t} \end{bmatrix} \tag{5.18}$$

where $N_1 = 1-\alpha$, $N_2 = \alpha$ and $\alpha = \dfrac{t-t_k}{\Delta t_k}$.

The derivatives for time of N_1 and N_2 are given as

$$\frac{d}{dt} \begin{bmatrix} N_1^t & N_2^t \end{bmatrix} = \begin{bmatrix} -\frac{1}{\Delta t_k} & \frac{1}{\Delta t_k} \end{bmatrix} \tag{5.19}$$

After substitution from equations (5.18) and (5.19), equations (5.17) take the form

$$\int_{t_k}^{t_k+\Delta t_k} \frac{1}{\bar{g}} \begin{bmatrix} 0 & 0 \\ 0 & H \end{bmatrix} \left[(1-\alpha) \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{tk} + \alpha \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{t_k+\Delta t_k} \right] dt$$

$$+ \int_{t_k}^{t_k+\Delta t_k} \frac{1}{\bar{g}} \begin{bmatrix} K & L \\ L^T & S \end{bmatrix} \left[-\frac{1}{\Delta t_k} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{tk} + \frac{1}{\Delta t_k} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{t_k+\Delta t_k} \right] dt \qquad (5.20)$$

$$= \int_{t_k}^{t_k+\Delta t_k} \frac{1}{\bar{g}} \begin{Bmatrix} \frac{df}{dt} + C \\ \bar{f} \end{Bmatrix} dt$$

The weighted residual method is employed to perform the integration for space dimensions. However, for the sake of economy, the point collocation is used in this instance. The matrices $[K],[L],[S],[H]$ and the force vectors need be evaluated only once per each time step, where $\alpha = \bar{\alpha}$ and $\bar{\alpha}$ may take any value from 0 to 1. For $\bar{\alpha} = 0$, $\bar{\alpha} = \frac{1}{2}$ and $\bar{\alpha} = 1$, the results are the same as for the standard finite difference method, i.e. forward difference (Euler), mid-difference (Crank-Nicholson) and backward difference respectively. Other values of α give rise to different schemes, for example:

(a) $\alpha = 0.66667$. This scheme represents a Galerkin type procedure with the weighting corresponding to the unknown function (see Zienkiewicz,[92] Ch. 21).

(b) $\alpha = 1 + (1/\Delta t) - 1/\ln(1+\Delta t)$. This is a logarithmic variation proposed by Sandhu[71] in which the coefficient α is a function of the size of the time step but not of the elapsed time.

(c) $\alpha = 1 + t_k/(t_{k+1} - t_k) - 1/\ln[1 + (t_{k+1} - t_k)/t_k]$ This is the logarithmic interpolation used by Hwang et al.[46] The coefficient α depends not only upon the elapsed time step $\Delta t = t_{k+1} - t_k$ but also upon the elapsed time t_k to the beginning of the interval.

Equations (5.20) may now be integrated, then divided by $\bar{g}$, and finally rearranged in the form

$$\begin{bmatrix} K & L \\ L^T & S+\bar{\alpha}H\Delta t_k \end{bmatrix}_{k,\alpha} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{t_k+\Delta t_k}$$
$$= \begin{bmatrix} K & L \\ L^T & S-(1-\bar{\alpha})H\Delta t_k \end{bmatrix}_{k,\alpha} \begin{Bmatrix} \bar{u} \\ \bar{p} \end{Bmatrix}_{t_k+\Delta t_k} + \begin{Bmatrix} \frac{df}{dt}+C \\ \bar{f} \end{Bmatrix}_{k,\alpha} \Delta t_k \qquad (5.21)$$

Equations (5.21) are formed for all internal nodes of the domain and those boundary nodes where pore pressure values and/or displacements are not prescribed. The number of equations is thus equal to the number of unknown variables.

The complete set of equations may be used in the time-stepping procedure outlined above to determine the values of $\{\bar{u}\}$ and $\{\bar{p}\}$ at any point in time relative to their initial values. In the non-linear case some or all of the matrices $[K]$, $[L]$, $[H]$, $[S]$ and vector $\{\bar{f}\}$, $\{f\}$ and $\{C\}$ are dependent on the values of the unknown, $\{\bar{u}\}$ and $\{\bar{p}\}$, so that theoretically iterations within each time step are required (except when $\bar{\alpha}=0$).

It can be easily verified that at steady-state conditions the coupling between the equilibrium equation and the flow equation disappears.

5.5 Oscillation and stability.

Numerical stability will now be considered for the case of free response. When the force terms are all zero,[63,92] equations (5.16) become

$$K\bar{u}+L\bar{p}=0 \tag{5.22a}$$

and

$$L^T\bar{u}+S\bar{p}+H\dot{\bar{p}}=0 \tag{5.22b}$$

Substituting for $\left\{\dot{\bar{u}}\right\}$ from equation (5.22a) into (5.22b) gives the governing equation for $\left\{\bar{p}\right\}$ as follows:

$$M\dot{\bar{p}}+H\bar{p}=0 \tag{5.23}$$

where

$$M=-L^TK^{-1}L+S$$

The standard solution for linear equations of this form is given by the sum of a set of modes of variation

$$\bar{p}=[a_1 \quad a_2 \quad \cdots][y_1 \quad y_2 \quad \cdots]^T \tag{5.24}$$

where

$$y_i=e^{\alpha_i t}$$

and

α_i=constant.

Substituting for $\{\bar{p}\}$ from equation (5.24) into equation (5.23), and pre-multiplying by $\{a\}^T$ gives the result

$$a^T M a \dot{y} + a^T H a y = 0 \tag{5.25}$$

It has been shown[92] that for eigenvalue solutions of equations such as (5.24)

$$a_i^T M a_j = a_i^T H a_j = 0, \; i \neq j \tag{5.26}$$

so that equation (5.25) reduces to a set of scalar independent equations of the form

$$m_i \dot{y}_i + n_i y_i = 0 \tag{5.27}$$

Application of the same argument to equation (5.21) also reduces these to a set of independent equations.

The elimination of $(\{\bar{u}\}_{t_k+\Delta t_k} - \{\bar{u}\}_{t_k})$ from equation (5.21) results in the following relationship

$$p_{tk+\Delta_{t_k}}\left[M + \bar{\alpha} H \Delta_{t_k}\right] = \bar{p}_{tk}\left[M - (1-\bar{\alpha}) H \Delta_{t_k}\right] \tag{5.28}$$

And on substituting for $\{\bar{p}\}$ by means of equation (5.24), and pre-multiplying by $\{y\}^T$, the following expression results:

$$(y_i)_{t_k+\Delta_k}[m_i+n_i\Delta_{t_k}]=(y_i)_{t_k}[m_i-n_i(1-\bar{\alpha})\Delta_{t_k}] \tag{5.29}$$

In order to prevent instability it is necessary that

$$|(y_i)_{t_k+\Delta_k}|\leq|(y_i)_{t_k}| \tag{5.30}$$

So that

$$|\lambda|\equiv\left[\frac{m_i-n_i(1-\alpha)\Delta t_k}{m_i+n_i\bar{\alpha}\Delta t_k}\right]\leq 1 \tag{5.31}$$

and hence

$$m_i-n_i(1-\bar{\alpha})\Delta t_k\geq-(m_i+n_i\bar{\alpha}\Delta t_k) \tag{5.32}$$

or

$$n_i(2\bar{\alpha}-1)\Delta t_k\geq-2m_i \tag{5.33}$$

The stability of the solution can be ensured by setting $\bar{\alpha}\geq\frac{1}{2}$ rather than checking equation (5.33) for every value of i.

The prevention of oscillation sets a more rigid requirement, i.e., $\lambda\geq 0$,

for which the only simple guarantee is setting $\bar{\alpha}=1$. The accuracy of the solution is then decreased, and some intermediate value of $\bar{\alpha}$, such as $\frac{2}{3}$, may be a better choice (see Zienkiewicz,[92] Ch. 21). In order to be sure of preventing oscillation it would again be necessary to check for every value of i. Also, it must be remembered that these arguments apply only to a linear analysis. In a non-linear analysis their usefulness would depend to a large extent on the validity of equation (5.24).

5.6. Numerical solution of saturated-unsaturated flow in porous media.

The finite element discretization of equations (3.38) and (3.54) is given below. The procedure is precisely the same as that for saturated flow given in Section 5.2, and is complicated only by the peculiar form of the average pressure:

$$\bar{p}=s_w p_w \tag{5.34}$$

and its time derivative

$$\frac{\partial \bar{p}}{\partial t}=S_w\frac{\partial p_w}{\partial t}+\frac{C_s}{\phi}p_w\frac{\partial p_w}{\partial t} \tag{5.35}$$

which has already been discussed in Section 2.6.

By inserting equation (5.35), the equilibrium equation (3.38) becomes

$$\begin{aligned}&\int_\Omega \delta\varepsilon^T D_T\frac{\partial \varepsilon}{\partial t}d\Omega-\int_\Omega \delta\varepsilon^T m(S_w\frac{\partial p_w}{\partial t}+\frac{C_s}{\phi}p_w\frac{\partial p_w}{\partial t})d\Omega\\&+\int_\Omega \delta\varepsilon^T D_T\frac{m}{3K_s}(S_w\frac{\partial p_w}{\partial t}+\frac{C_s}{\phi}p_w\frac{\partial p_w}{\partial t})d\Omega-\int_\Omega \delta\varepsilon^T D_T c\,d\Omega\\&-\int_\Omega \delta\varepsilon^T D_T\frac{\partial \varepsilon_0}{\partial t}d\Omega-\frac{\partial f}{\partial t}=0\end{aligned} \tag{5.36}$$

Equation (5.36) is then coupled with the continuity equation (3.79) for the case of saturated-unsaturated flow. A weak form of equation (3.79) is again obtained by means of Green's theorem as in Section 5.3.

The displacements and pore pressures are expressed in terms of their nodal values $\{\bar{u}\}$ and $\bar{p}_w$ as shown in equations (5.9a)-(5.9c). The finite element discretization in space again yields the following ordinary differential equations in time:

$$K\frac{d\bar{u}}{dt}+\bar{L}\frac{d\bar{p}_w}{dt}-C-\frac{df}{dt}=0 \tag{5.37}$$

and

$$\bar{H}\bar{p}_w+\bar{S}\frac{dp_w}{dt}+T^T\frac{d\bar{u}}{dt}-\bar{f}=0 \tag{5.38}$$

where in equation (5.37) only $[\bar{L}]$ changes with respect to equations (5.13a)-(5.13d) and is defined as

$$\bar{L}=\int_\Omega B^T S_r(m-\frac{mD_r}{3K_s})\bar{N}d\Omega+\int_\Omega B^T\frac{C_r}{\phi}(m-\frac{mD_r}{3K_s})\bar{N}\bar{p}_c\bar{N}d\Omega \tag{5.39}$$

The other matrices are defined by

$$\bar{H}=\int_\Omega(\nabla\bar{N})^T\frac{k_{rw}}{\mu_w}k\nabla\bar{N}d\Omega \tag{5.40a}$$

$$\bar{S}=\int_\Omega\bar{N}^T\bar{s}\bar{N}d\Omega+\int_\Omega\bar{N}^T\overline{sN}\bar{p}_w\bar{N}d\Omega$$

where

$$\bar{s} = \overline{C_s} + \phi \frac{S_w}{K_w} + (S_w)^2 \left(\frac{1-\phi}{K_s} - \frac{1}{(3K_s)^2} m^T D_T m \right) \tag{5.40b}$$

and

$$\bar{s} = \frac{S_w \bar{C}_s}{\phi} \left(\frac{1-\phi}{K_s} - \frac{1}{(3K_s)^2} m^T D_T m \right)$$

$$T = \int_\Omega \overline{N}^T S_w \left(m^T - \frac{m^T D_T}{3K_s} \right) B d\Omega \tag{5.40c}$$

and finally

$$\bar{f} = -\int_\Gamma N^T q d\Gamma - \int_\Omega \frac{S_w}{3K_s} m^T D_T C d\Omega - \int_\Omega (\nabla \overline{N})^T \frac{k_w k}{\mu_w} \nabla \rho g h d\Omega \tag{5.40d}$$

For convenience, the equations are written in the following form:

$$\begin{bmatrix} 0 & 0 \\ 0 & H \end{bmatrix} \begin{Bmatrix} \bar{u} \\ \bar{p}_w \end{Bmatrix} + \begin{bmatrix} K & \bar{L} \\ T^T & S \end{bmatrix} \frac{d}{dt} \begin{Bmatrix} \bar{u} \\ \bar{p}_w \end{Bmatrix} = \begin{Bmatrix} \frac{df}{dt} + C \\ \bar{f} \end{Bmatrix} \tag{5.41}$$

The similarity to equation (5.16) is worthy of further attention. A state of symmetry is restored to equation (5.41), except for non-associated plasticity, if $C_s p << S_w \phi$. The validity of this assumption should be checked for each particular porous medium. Otherwise, a partitioned solution may be used to restore symmetry (see Lewis and Schrefler,[52] Ch. 7).

The matrices $[\overline{H}], [\overline{S}], [\overline{L}]$ and $[T]$ and the vector $\{f\}$ are now non-linear due to the presence of the saturation, the specific moisture content, the relative permeability and the pressure. The integration of equations (5.38), (5.40a), (5.40b), (5.40c) and (5.40d) is carried out numerically via

the Gauss quadrature formulae. The functions S_w, C_s and k_{rw} have to be evaluated at the Gauss points as a function of the pressure p.

In a similar manner to the solution derived by Neuman[62], this approach calculates the location of the free surface, where $p_w = 0$, simply as an isobar. This surface also happens to separate the saturated from the unsaturated zones. The technique is different from the classical approach where this surface is treated as a moving material boundary.

5.6 Concluding remarks.

Various solution procedures for the governing equations of multiphase flow in a deforming porous medium have been discussed in this chapter. Solutions for the governing equations of one-phase flow and saturated-unsaturated flow in such a medium by means of the finite element method have been presented in detail.

Before these solutions can be applied to the investigation of consolidation in the general case, and surface subsidence due to withdrawal of underground fluids, the stiffness and flow matrices must be determined.

Chapter 6

Procedures and principles of continuum model for consolidation due to pumpage

Dr. Walter Kjellman, of the Swedish Geotechnical Institute, introduced the consolidation of weak soils by vacuum in 1952. Until recently, however, the method had been applied in a predictable manner only to extremely soft soils and large-scale projects. The vacuum method was first used in China, and after scattered efforts aiming at practical applications in the U.S. and elsewhere, it has now evolved into reliable technology. Special credit goes to Jean-Marie Cognon, co-founder of Menard Soltraitement of France, who solved methodology problems associated with this technique by rethinking basic theoretical principles in the late 1980s. Nowadays, the patented process is known as the Menard Vacuum Consolidation method.[18,19,20,27]

This chapter begins with a brief description of the theoretical principles of vacuum-induced consolidation. The procedures and principles of existing methods for consolidation settlements, which were presented in the previous chapters, will then be further developed. A new approach will be presented for the analysis of the continuum model for consolidation by pumpage, namely, the steps required for a linear equation solution method for separate fixed boundary conditions.

6.1 Theoretical aspects of vacuum-induced consolidation.

The basic procedure of the Menard vacuum system consists in removing atmospheric pressure from a confined, sealed medium of soil to be consolidated while maintaining the vacuum for a pre-determined period of time, as illustrated in Figure 6.1.

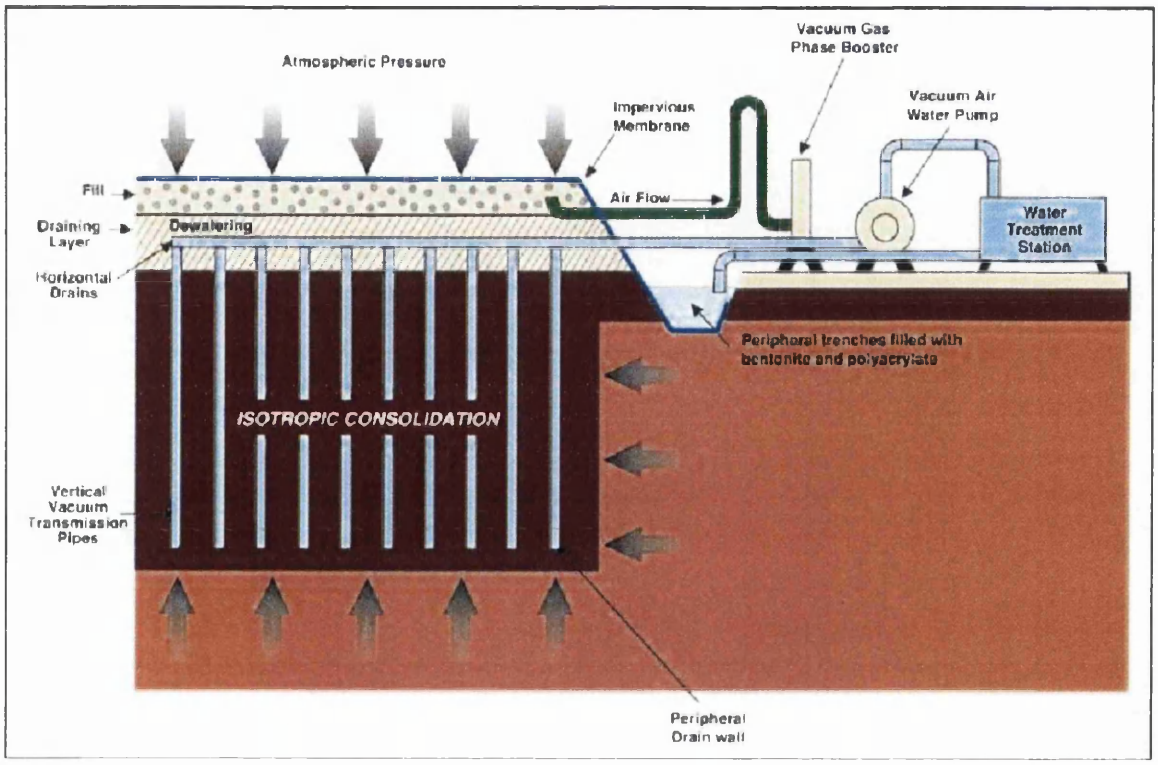


Figure 6.1 Typical cross-section of Menard Vacuum Consolidation[aa]

Technological problems associated with this method include:

- maintaining an effective drainage system under the membrane that expels water and air throughout the pumping duration
- keeping non-water saturated medium below the membrane
- maintaining an effective level of vacuum (preferably 30% of the atmospheric pressure)
- maintaining a leakproof system, in particular at the pump / membrane connections, as well as over the whole membrane area
- anchoring and sealing the system at the periphery
- reducing lateral seepage towards the vacuum area.

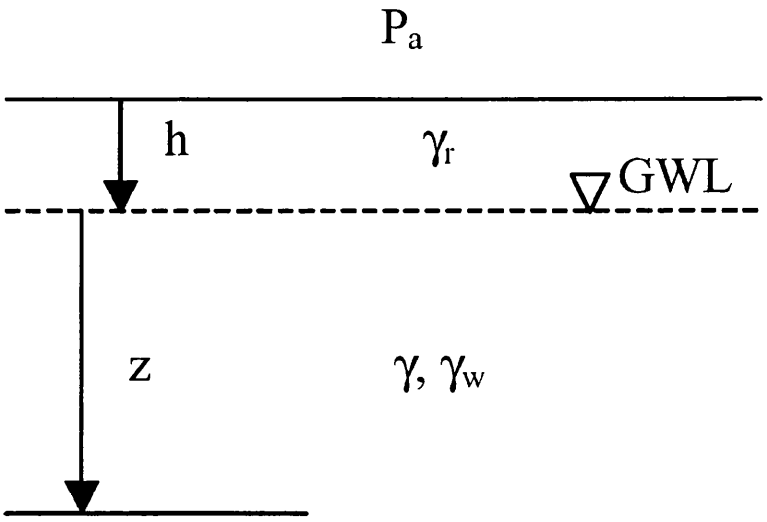
The key objectives of the Menard Vacuum Consolidation system are:

- to enable immediate loading under unstable conditions (soft soil is prone to circular failures under embankment load); and
- to obtain a "pre-aging" of the compressible soil layers by having primary and secondary settlements occur before construction.

Before explaining how these objectives can be met, it is first necessary to describe the effect of vacuum on saturated soil.

6.1.1 The atmospheric pressure.

The atmospheric pressure P_a is generally disregarded in soil stress calculations, but atmospheric pressure should be taken into account to understand the effects of the Menard vacuum system. The soil stress state before vacuum is described by the following equations with atmospheric pressure:



$$\begin{aligned} \sigma_T &= z\gamma + h\gamma_r + P_a = \sigma_t + P_a \\ u_T &= z\gamma_w + P_a = u_t + P_a \\ \sigma_i^{'} &= \sigma_T - u_T = \sigma_t - u_t = z\gamma^{'} + h\gamma_r \end{aligned} \qquad (6.1)$$

where

T refers to stress with atmospheric pressure
t is stress without atmospheric pressure
i indicates initial state

$\sigma_i^{'}$ is the effective stress before vacuum.

After vacuum has been applied, total stress σ_T does not change, whereas u_T becomes $u_t = z\gamma_w + 0$ so the final effective stress (during vacuum) becomes

$$\sigma'_f = \sigma_T - u_T = \sigma'_i + P_a \tag{6.2}$$

and thus, taking into account the efficiency of the vacuum pump:

$$\Delta\sigma' = nP_a \tag{6.3}$$

Vacuum increases effective stress by one atmospheric in any direction because this increment is transmitted through the water phase, which is by essence isotropic.

6.1.2 Pre-aging method.

In order to obtain less than 10 cm of settlement in the decade after the Menard vacuum system has been installed, it is necessary to reach the settlement target illustrated in Figure 6.2.

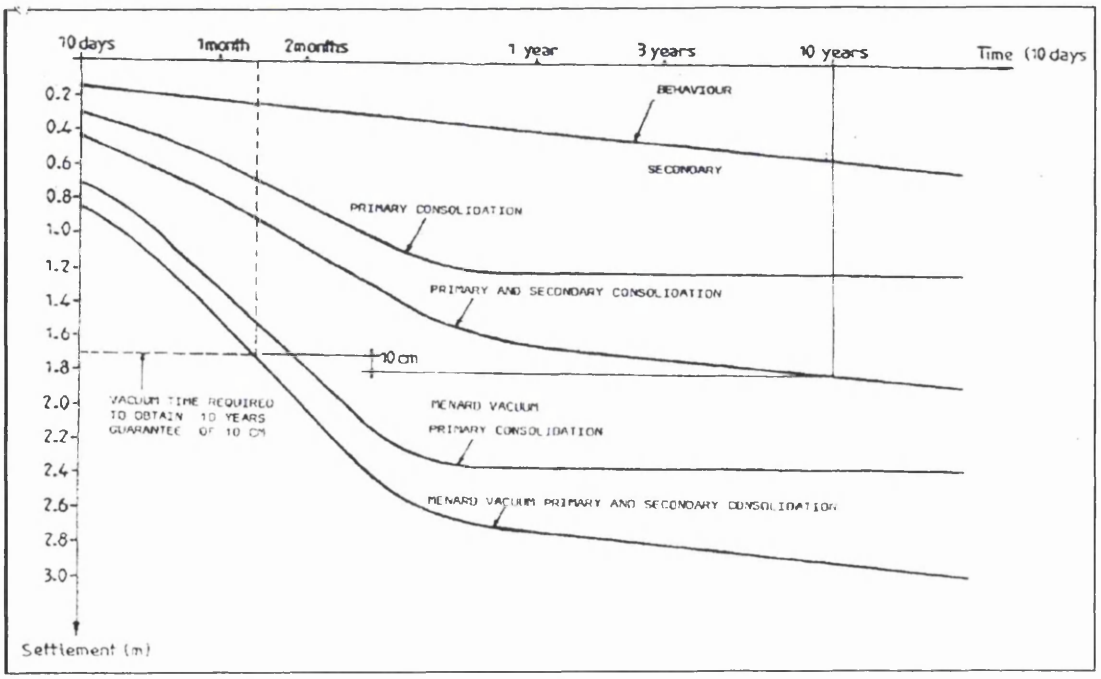


Figure 6.2 Settlement objective of Menard vacuum system

The desired ratio reduction target during treatment corresponds to $\Delta e = 100\%$ primary consolidation + 10 years of secondary settlement under the combined operational design load with a maximum allowable settlement of 100 mm over 10 years.

For secondary settlement, "pre-aging" involves increasing the effective stress beyond the normal consolidated state. Based on Bjerrum theory, this is illustrated in Figure 6.3.

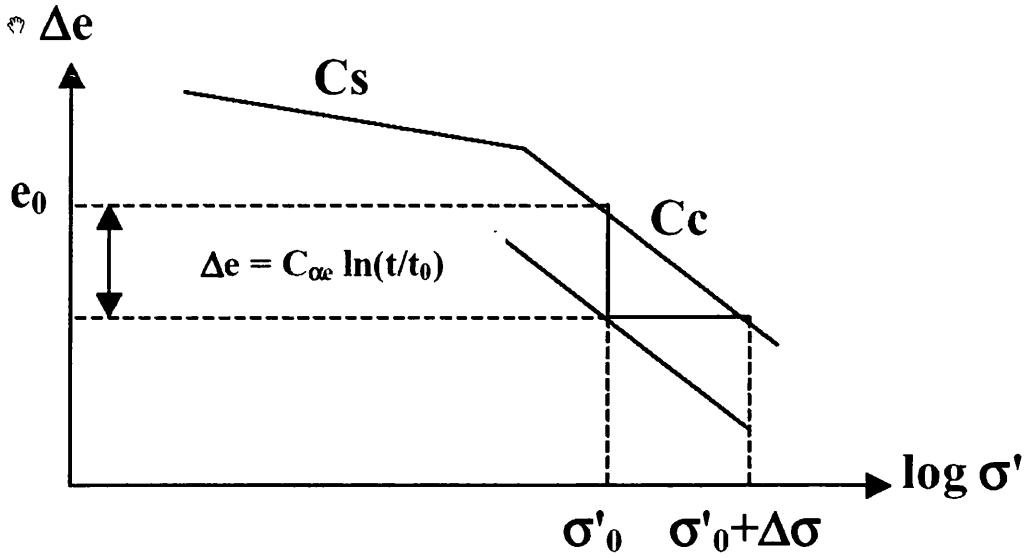


Figure 6.3 Pre-aging and secondary settlement

Utilizing the design loads and the results of preliminary soil investigation, the pre-aging objective is converted into a settlement target T_{th} to be reached during the vacuum pumping period, as in Figure 6.2. As there are always discrepancies between consolidation theory and actual measurements, it is necessary to calibrate the model with settlement monitoring results. Because soils are actually aggregate materials, we cannot rely on continuum mechanical theory to calculate the complex phenomena of soils. Final settlement can be assessed *in situ* using the Asaoka method [2] for the measurement of ongoing settlement during vacuum consolidation. This method takes into account the theories underlying Terzaghi's vertical and Baron's radial consolidation. [4] Once the Asaoka settlement is known under a given load, it is possible to

compare the new target value T_{Asa} with the theoretical target T_{th}.

Deep settlement gauges must also be installed to ensure that every layer reaches the target void ratio. This can be summarized as the following equation, for each layer of soil:

$$\beta = \frac{\Delta T_{Asaoka}}{\Delta T_{theory}} = \frac{(\frac{C_c}{1+e_0})_{actual}}{(\frac{C_c}{1+e_0})_{soilinvestigation}} \tag{6.4}$$

where C_c is the compression index.

In addition, the coefficient of radial consolidation C_r can be computed using the Asaoka method, as illustrated in Figure 6.4. It is then compared with the coefficient determined from preliminary soil investigation results and gives a second assessment of the consolidation rate for each layer.

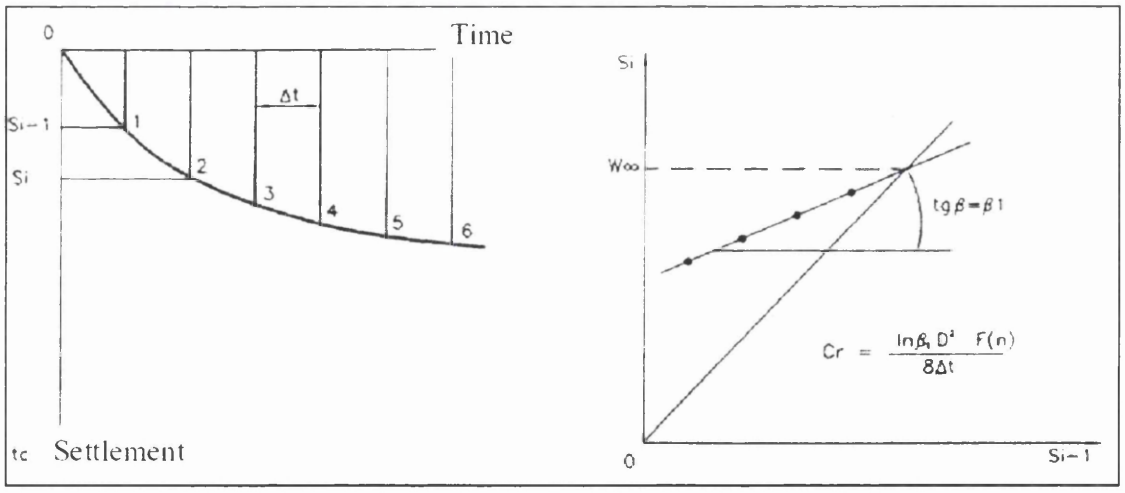


Figure 6.4 Asaoka construction and interpretation

6.1.3 Stress path, horizontal displacement and stress influence in depth.

In the case of surcharge consolidation, as shown in Figure 6.5, the stress path on the (p',q') diagram follows the AB curve with a possibility of slope failure in the event that point B crosses the failure line. From

there, consolidation follows the BC curve. Line AD corresponds to perfect oedometric consolidation. In the case of vacuum, the stress path simply follows the AE line. The stress path remains below the K_0-line and is subject to quasi isotropical effective stress. When surcharge is associated with the Menard vacuum system, superposition of states shall be applied. Horizontal displacement is limited to the corresponding radial consolidation of the soil within the drain influence. The volume of soil influenced by the peripheral drains, in the case of a semi-infinite medium, happens up to one foot of the lateral displacement can be found in very soft clays.

The Menard vacuum system has been shown [47] to be effective and reasonably economical when installed at a depth of 30 m. Plain surcharge methods, however, are less effective at this depth because of the stress reduction factor. For instance, based on elastic theory for an isotropic, semi-infinite medium, the load of an embankment 18 m wide is reduced by almost half at its centre at an 18 m depth, and by more than 70% at its edge at the same depth.

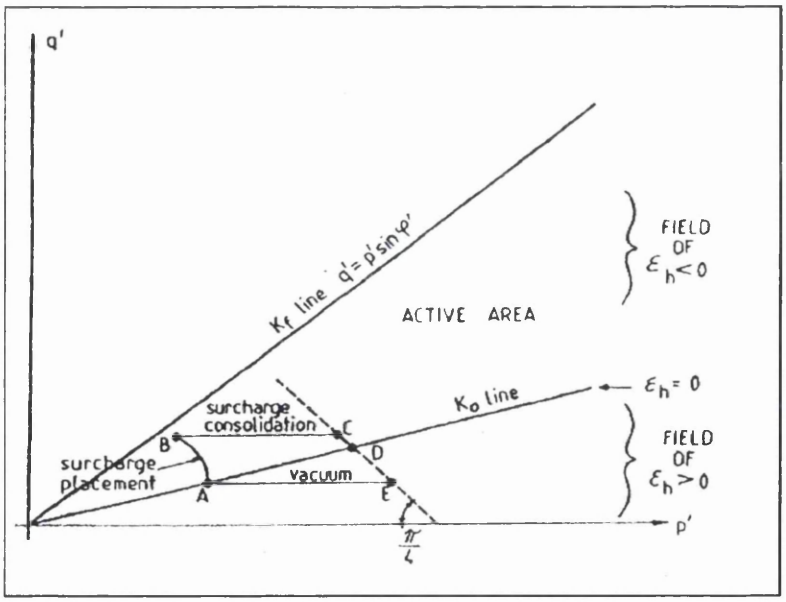


Figure 6.5 Vacuum on $((p'-q'))$ diagram

6.2 The finite element continuum model of consolidation by pumpage.

Most analytical methods of promoting weak ground settlements are

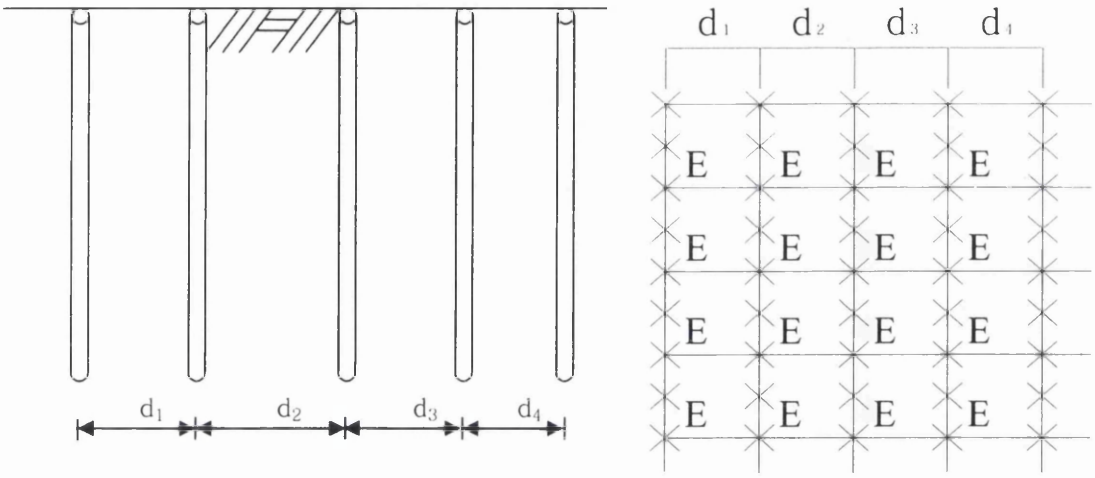
based on Terzaghi's one-dimensional consolidation theory or Baron's vertical drain, in conjunction with field measurements. These theories are easy to apply and field measurements supplement the gap between theory and actual displacements. But soil properties are complex, so it is not easy to get exact results of natural states when applying the principles of continuum mechanics. More precise results can be obtained by starting with Biot's three-dimensional consolidation theory.
In this thesis, the problem of promotion of consolidation settlements is analysed by the finite element method as applied to the changing material properties of soil. It is difficult to model the installed material, whether pack, prefblicated vertical drain, or Menard tube, and even when modelled as accurately as possible, the finite element method did not produce a solution for the problem caused by diverging at the parts of the installed material. For example, the diameter of installed material as pack may be 50 mm, and the thickness of prefablicated drain material just a few mm, and the Menard tube also 50 mm. Given a distance of approximately 1000 mm from one installed hole to another, analysis of the modelled problem by the finite element method is made difficult because of the proximity of a small element and a large element at the site of the problem.

None of the existing methods deal with changing the pore pressures of installed material, when dealing with the pack at pack drain method, material for prefablicated drain, and Menard tube at the Menard vacuum drain method. For this reason, a new approach is needed in the analysis of the continuum model for consolidation by pumpage. The continuum theories described in Chapters 3, 4 and 5 must be further developed with new ideas for treating interior boundary conditions and new linear equations.This thesis presents an innovative linear equation solution method for separate fixed boundary conditions, with changing pore pressures at installed material being treated as boundary conditions of the interior part.

6.2.1 The idea of boundary conditions for the interior part.

Analysis of the drain material part is needed to treat pore pressure boundary conditions for the interior part of the analysis domain, as

shown in Figure 6.6.



(a) field installed vacuum tube (b) interioer boundary condition

Figure 6.6 Fixed boundary conditions for pore pressure of interior part

where $d_1, d_2, \cdots$= practical distance of pumpage drain hole

E = each element

×= prescribed boundary conditions of pore pressure.

The interior boundary condition differs from the general or outer boundary conditions found outside the body to be analysed. It is a fixed boundary condition that contains the interior part of the body, and a special linear equation solving method is needed to treat the fixed boundary condition separately. The proposed solving method for the interior fixed boundary condition is a more useful tool for analysing the problem of consolidation due to pumpage because calculation does not involve changing the material properties of the analysis domain. The method of changing materials is a reason for errors to occur when using finite element method simulation.

In the proposed solution method for fixed boundary condition, the interior boundary condition must be changed to a non-fixed (free) condition after pumpage has finished. As well, the consolidation analysis must be calculated by the changed boundary condition, so that another linear equation solver with separated boundary condition is needed.

6.2.2 A blocked profile solver of the linear equation solution with separate fixed boundary conditions.[48]

The proposed consolidation analysis by pumpage must solve the linear equation system for treating fixed boundary condition separately because the boundary conditions for pore pressures may be changed with every time step.

Numerical analysis of partial differential equations is required to transform these equations to a linear equation system.[9] Numerical analyses of the finite difference, finite element, and boundary element methods of the partial differential equation are needed to develop a solution system of linear equations. Until now, two kinds of solving methods have been used - the direct method and the iterative solver.[1,5,42,52,56,84,85,95] The conjugate gradient[42,84] is an iterative solver for symmetric types of global stiffness matrix that has been shown to be efficient when using a multi-processor, however, is that it needs longer CPU time than a direct solver to get exact results.

Many kinds of direct solving methods of the linear equation system have been developed, with frontal and profile methods being the most popular to solve symmetric and unsymmetric linear equations. The frontal method is useful for solving unsymmetric linear equations and fixed boundary conditions, but the profile method is faster for symmetric ones because its large front width can deal with a large number of equations.

Manoj and Bhattacharyya[56] developed the blocked unsymmetric solver. A few years later, Demmel and others[30] developed a profile solver for symmetric and unsymmetric solver with pivoting written C language, but this linear equation system could not be applied to finite element analysis. None of these profile solvers[5,30] described fixed boundary conditions separately.

The proposed solving method of this thesis is a symmetric and unsymmetric linear equation system with separated routines for fixed boundary conditions and direct application to finite element analysis.

6.3 Practical considerations of the continuum finite element consolidation modelling due to pumpage.

When modelling the field problem by finite element analysis, it is more convenient to fit the meshing sizes at the same intervals as the Menard tube, so that many numbers of nodal points and elements are created. Finite element modelling for the field problem is outlined in next sections.

6.3.1 The meshing sizes for the field problem.

The mesh size of elements must fit within the distance of the Menard tube, as shown in Figure 6.7, in order to apply the fixed boundary conditions of pore pressures to the continuum model used by the finite element method.

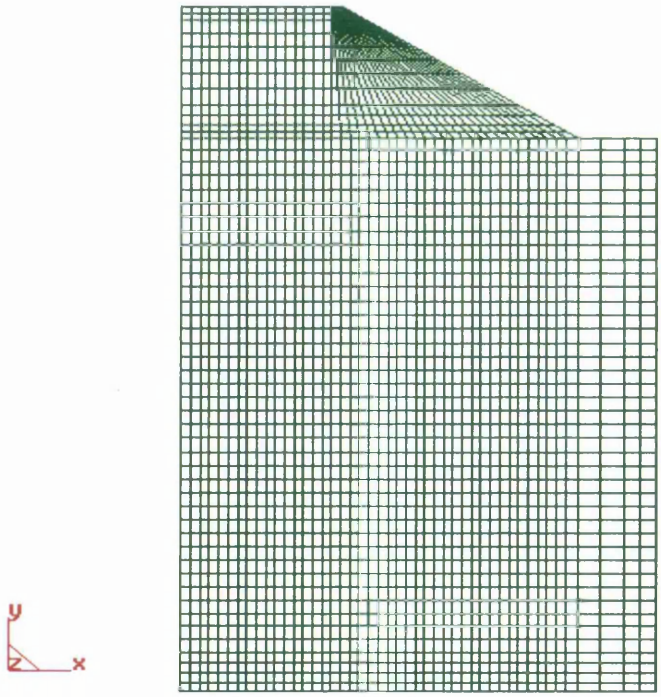


Figure 6.7 Finite Element modelling of the field problem.

This modelling is required for the solving method of the linear equation system with separated routine of fixed boundary conditions.

6.3.2 Material properties of the continuum consolidation modelling by pumpage.

Material properties are very important for analysis of exact differential equations or numerical methods, such as finite difference, boundary element or the finite element method. Properties are determined by experiment, but it is not possible to get exact results for all material properties of soils because the soil samples are too much smaller than actual field materials. In this thesis, some experimental material properties are used. The next section describes the solution method of the linear equation system with separated fixed boundary condition routine for use the prior solution of the problem.

6.4 Solution algorithm of symmetric and unsymmetric linear equation system with separated fixed boundary condition routine for fixed pore pressure boundaries.

If the governing partial differential equations have terms of $\frac{\partial}{\partial x}$ then the element stiffness matrix is unsymmetric.[50] If the element matrices are unsymmetric, the global stiffness matrix is also unsymmetric. A profile solver can be used for the solution algorithm of symmetric or unsymmetric linear equation systems.

6.4.1 Algorithm of symmetric and unsymmetric profile solver.

The first step is to write LU decomposition procedure of symmetric and unsymmetric global matrices.

The unsymmetric system of linear equation is shown as follows:

$$\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_n \end{Bmatrix} \tag{6.5}$$

Equation (6.5) must be solved by the profile method, if the matrix $[K]$ is of this type. The general Gauss elimination method using the $[L][S]$ factorization procedures is outlined next. The $[S]$ matrix is generally the same as $[L][U]$ decomposition.

6.4.2 LU decomposition of symmetric and unsymmetric global matrix.

Decomposition of a general matrix can be developed to obtain all of the terms of $[L]$ and $[S]$ from the original global stiffness matrix$[K]$. All terms of $[L]$ and $[S]$ are stored in the original global stiffness matrix $[K]$ as follows:

$$\begin{bmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ l_{31} & l_{32} & 1 & & & \\ \cdot & \cdot & \cdot & 1 & & \\ \cdot & \cdot & \cdot & \cdot & 1 & \\ l_{n1} & \cdot & \cdot & \cdot & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdot & \cdot & s_{1n} \\ & s_{22} & s_{23} & \cdot & \cdot & s_{2n} \\ & & s_{33} & \cdot & \cdot & s_{3n} \\ & & & \cdot & \cdot & \cdot \\ & & & & s_{n-1,n-1} & s_{n-1,n} \\ & & & & & s_{n,n} \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & \cdots & k_{1n} \\ k_{21} & k_{22} & k_{23} & \cdots & k_{2n} \\ k_{31} & k_{32} & k_{33} & \cdots & k_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ k_{n1} & k_{n2} & k_{n3} & \cdots & k_{nn} \end{bmatrix} \tag{6.6}$$

Next, the procedure of decomposition is obtained.

$n = 1$:

$1 \bullet s = k_{11} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots s_{11} = k_{11}$

$n = 2$:

$l_{21} s_{11} = k_{12} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots l_{21} = k_{21} (s_{11})^{-1}$

$l_{21} s_{12} + s_{22} = k_{22} \ldots\ldots\ldots\ldots\ldots\ldots s_{22} = k_{22} - l_{21} s_{12}$

$n = 3$:

$l_{31} s_{11} = k_{31} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots l_{31} = k_{31} (s_{11})^{-1}$

$l_{31} s_{12} + l_{32} s_{22} = k_{32} \ldots\ldots\ldots\ldots\ldots l_{32} = (k_{32} - l_{31} s_{12})(s_{22})^{-1}$

$1 \bullet s_{13} = k_{13} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots s_{13} = k_{13}$

$l_{21} s_{13} + s_{23} = k_{23} \ldots\ldots\ldots\ldots\ldots\ldots s_{23} = k_{23} - l_{21} s_{13}$

$l_{31} s_{13} + l_{23} s_{32} + s_{33} = k_{33} \ldots\ldots\ldots s_{33} = k_{33} - l_{31} s_{13} - l_{32} s_{23}$

For any value of S

$$l_{ni} = \left(k_{ni} - \sum_{m=1}^{i-1} l_{nm}\, s_{mi} \right) s_{ii}^{-1} \qquad i = 1, 2, \ldots, n-1$$

$$s_{jn} = k_{jn} - \sum_{m=1}^{j-1} l_{jm} s_{mn} \qquad j = 1, 2, \ldots, n \qquad (6.7)$$

With matrices $[L]$ and $[S]$ being stored in $[K]$ as shown in equation (6.6), the previous algorithm becomes:

$l = 2, 3, \ldots, n$

  $i = 1, 2, \ldots, l-1$

    $m = 1, 2, \ldots, i-1$

      $k_{li} = k_{li} - k_{lm} k_{mi}$ row of L

      $k_{il} = k_{il} - k_{im} k_{ml}$ column of S

    $k_{li} = k_{li} k_{ii}^{-1}$ normalize row L

  $m = 1, 2, \ldots, l-1$

$k_{ll} = k_{ll} - k_{lm} k_{ml}$ diagonal term

If the global matrix is symmetric, the lower part of the global stiffness matrix $[K]$ is not used during decomposition. Note that this algorithm does not contain fixed boundary conditions. Treatment of fixed boundary conditions will be described separately.

6.4.3 Reduction of load vector and back-substitution.

As already discussed, the decomposition of the global stiffness matrix$[K]$ forms the main part of the equation solution. Once the$[L]$, $[S]$ factors of $[K]$ have been obtained, the load vector is reduced, using the lower part of the matrix as shown in the next equations.

$$i = 2,3,.....n$$
$$f_i = f_i - \sum_{j=1}^{i-1} k_{ij} f_j \qquad (6.8)$$

After reduction has been carried out, the solution vector can be calculated by a back-substitution procedure using the upper part of the decomposed matrix as in equation (6.9).

$$f_n = k_{nn}^{-1} f_n$$
$$i = n-1, n-2, ..., 1 \qquad (6.9)$$
$$f_i = k_{ii}^{-1}\left(f_i - \sum_{j=i+1}^{n} k_{ij} f_j \right)$$

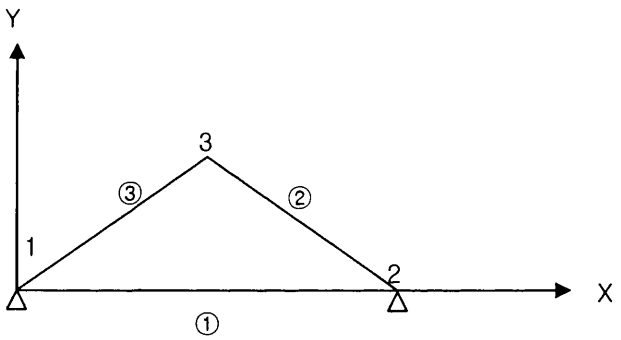
The vector $\{F\}$ is the value of solution as vector $\{U\}$ generally. As yet, the solving procedures do not include the boundary condition part.

6.4.4 Blocked profile scheme of symmetric and unsymmetric linear equation system.[5,48,85]

A blocked profile scheme consists of assigning equation numbers for the degrees of freedom per node, element connectivity, column height of every equation, and diagonal address of each equation. In addition, the global stiffness matrix must be divided into many blocks which depend on computer memory or user-defined memory size.

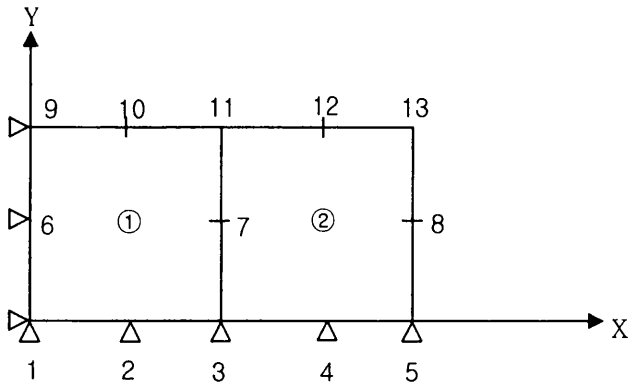
6.4.4.1 Assign equation number.

The number of nodal points must be assigned for the profile solution method. These can be calculated by the nodal degrees of freedom as in Figure 6.8.



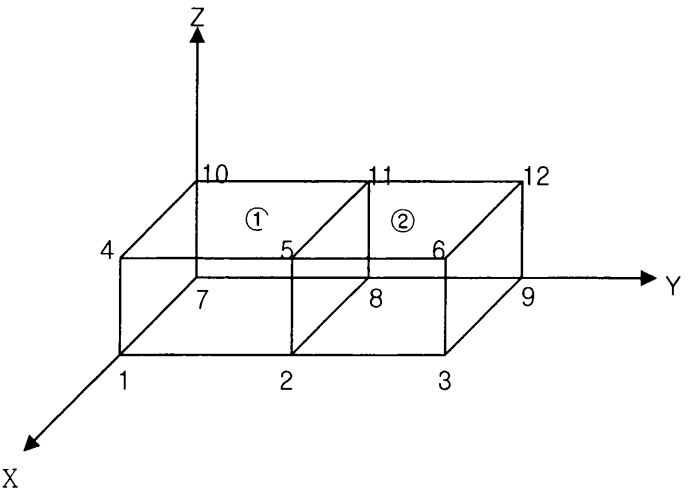
Node number	1		2		3	
Degree of freedom	1	1	0	1	0	0
Global Eq. number	-1	-2	3	-4	5	6

(a) Two-dimensional bar elements



Node number	1		2		3		4		5		6		7	
Degree of freedom	1	1	0	1	0	1	0	1	0	1	1	0	0	0
Global Eq. nu.mber	-1	-2	3	-4	5	-6	7	-8	9	-10	-11	12	13	14
Node number	8		9		10		11		12		13			
Degree of freedom	0	0	1	0	0	0	0	0	0	0	0	0		
Global Eq. number	15	16	-17	18	19	20	21	22	23	24	25	26		

(b) Two-dimensional 8-noded plane strain elements



Node	1	2	3	4

number												
Degree of freedom	0	1	1	0	0	1	0	1	1	0	1	0
Global Eq. number	1	-2	-3	4	5	-6	7	-8	-9	10	-11	12
Node number	5			6			7			8		
Degree of freedom	0	0	0	0	0	0	1	0	1	1	0	1
Global Eq. number	13	14	15	16	17	18	-19	20	-21	-22	23	-24
Node number	9			10			11			12		
Degree of freedom	1	0	1	1	1	0	1	0	0	1	0	0
Global Eq. number	-25	26	-27	-28	-29	30	-31	32	33	-34	35	36

(c) Three-dimensional 8-noded solid elements

Figure 6.8. Equation numbers of every nodal point

In Figure 6.8 and hereafter, the negative numbers of the equation are used to treat fixed boundary conditions separately.

6.4.4.2 Element connectivity.

Element connectivity arrays will be assigned by the nodal connectivity of each element and degrees of freedom per node, as in Figure 6.9 (the assigned model of Figure 6.8).

Element number	1				2				3			
Nodal Connectivity	1		2		2		3		3		1	
Global Eq. number	-1	-2	3	-4	3	-4	5	6	5	6	-1	-2

(a). Two-dimensional bar elements

Element number	1															
Nodal Connectivity	1		2		3		7		11		10		9		6	
Global Eq. number	-1	-2	3	-4	5	-6	13	14	21	22	19	20	-17	18	-11	12
Element number	2															
Nodal Connectivity	3		4		5		8		13		12		11		7	
Global Eq. number	5	-6	7	-8	9	-10	15	16	25	26	23	24	21	22	13	14

(b). Two-dimensional 8-node plane strain elements

Element number	1											
Nodal Connectivity	1			2			8			7		
Global Eq. number	1	-2	-3	4	5	-6	-22	23	-24	-19	20	-21
Element number	1											
Nodal Connectivity	4			5			11			10		
Global Eq. number	10	-11	12	13	14	15	-31	32	33	-28	-29	30
Element number	2											
Nodal Connectivity	2			3			9			8		
Global Eq. number	4	5	-6	7	-8	-9	-25	26	-27	-22	23	-24
Element number	2											
Nodal Connectivity	5			6			12			11		

Global Eq. number	13	14	15	16	17	18	-34	35	36	-31	32	33

(c). Three-dimensional 8-node brick elements

Figure 6.9. Examples of element nodal connectivity and equation numbers (negative numbers are fixed boundary parts)

6.4.4.3 Column height of each equation.

The element pattern of a typical stiffness matrix is presented in Figure 6.10. In this section, the proposed storage scheme and addressing procedure will be derived, to be used with an active column solver (profile solver). The matrix is symmetric or unsymmetric, we choose to store and work on above and below including diagonal. In addition, note that elements (i,j) of $[K]$ (i.e., k_{ij}) are zero for $j > i + m_k$. The value m_k is known as the half-bandwidth of the matrix. Defining by m_i the row number of the first non-zero element in column i (Figure 6.10), the variable $m_i, i = 1,...,n$: i.e., m_k is equal to the maximum difference of global degrees of freedom pertaining to any one of the finite elements in the mesh. In many finite element analyses, the column heights vary with i, and it is important that all zero elements outside the skyline not be included in the equation solution.

6.4.4.4 Diagonal elements of the global stiffness matrix.

With the column heights of a global stiffness matrix defined, all elements below the skyline of $[K]$ can be stored as a one-dimensional array of $[A]$ matrix; i.e., the active columns of $[K]$ including the diagonal elements are stored in the $[A]$ matrix. In addition to $[A]$, it is necessary

to define an array MAXAJ, which stores the addresses of diagonal elements of $[K]$ in $[A]$; i.e., the address of the diagonal element of $[K]$, k_{ii}, in $[A]$ is MAXAJ(I), MAXAJ(I)+ 1, MAXAJ(I)+ 2,$\cdots$,MAXAJ(I+ 1)−1. As in Figure 6.10, when using this storage scheme of $[K]$ in $[A]$, it follows that:

$$[K]=\begin{bmatrix} k_{11} & k_{12} & & k_{14} & & & & k_{18} \\ & k_{22} & k_{23} & 0 & & & & k_{28} \\ & & k_{33} & k_{34} & & k_{36} & & k_{38} \\ & & & k_{44} & k_{45} & k_{46} & & k_{48} \\ & & & & k_{55} & k_{56} & & k_{58} \\ & & & & & k_{66} & k_{67} & 0 \\ \textit{symmetric} & & & & & & k_{77} & k_{78} \\ & & & & & & & k_{88} \end{bmatrix}$$

$$[A]=\begin{bmatrix} a_1 & a_3 & & a_9 & & & & a_{25} \\ & a_2 & a_5 & a_8 & & & & a_{24} \\ & & a_4 & a_7 & & a_{15} & & a_{23} \\ & & & a_6 & a_{11} & a_{14} & & a_{22} \\ & & & & a_{10} & a_{13} & & a_{21} \\ & & & & & a_{12} & a_{17} & a_{20} \\ \textit{symmetric} & & & & & & a_{16} & a_{19} \\ & & & & & & & a_{18} \end{bmatrix}$$

a) An example of symmetric stiffness matrix.

$$[K]=\begin{bmatrix} k_{11} & k_{12} & & k_{14} & & & & k_{18} \\ k_{21} & k_{22} & k_{23} & 0 & & & & k_{28} \\ & k_{32} & k_{33} & k_{34} & & k_{36} & & k_{38} \\ k_{41} & 0 & k_{43} & k_{44} & k_{45} & k_{46} & & k_{48} \\ & & & k_{54} & k_{55} & k_{56} & & k_{58} \\ & & k_{63} & k_{64} & k_{65} & k_{66} & k_{67} & 0 \\ & & & & & k_{76} & k_{77} & k_{78} \\ k_{81} & k_{82} & k_{83} & k_{84} & k_{85} & k_{86} & k_{87} & k_{88} \end{bmatrix}$$

$$[A]=\begin{bmatrix} a_1 & a_3 & & a_9 & & & & a_{25} \\ b_3 & a_2 & a_5 & a_8 & & & & a_{24} \\ & b_5 & a_4 & a_7 & & a_{15} & & a_{23} \\ b_9 & b_8 & b_7 & a_6 & a_{11} & a_{14} & & a_{22} \\ & & & b_{11} & a_{10} & a_{13} & & a_{21} \\ & & b_{15} & b_{14} & b_{13} & a_{12} & a_{17} & a_{20} \\ & & & & & b_{17} & a_{16} & a_{19} \\ b_{25} & b_{24} & b_{23} & b_{22} & b_{21} & b_{20} & b_{19} & a_{18} \end{bmatrix}$$

b) An example of unsymmetric stiffness matrix.

Figure 6.10.Storage scheme used for a typical stiffness matrix, column heights and diagonal addresses.

The element of matrix $[A]$ is allocated at the same position of stiffness matrix $[S]$ to store vector form.

The column height numbers of elements for the stiffness matrix are shown from the diagonal element to skyline.

Column heights of equations, except the diagonal part of the global stiffness matrix, are calculated as in section 6.4.4.3 and are shown as follows:

equation number	1	2	3	4	5	6	7	8
column height	0	1	1	3	1	3	1	7

Next, the diagonal addresses of a typical stiffness matrix are assigned as follows:

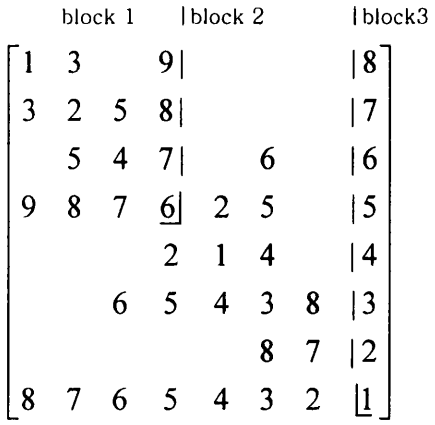
equation number	1	2	3	4	5	6	7	8	
diagonal address	1	2	4	6	10	12	16	18	26

6.4.5 A block scheme for the symmetric and unsymmetric linear equation system.

The block scheme is more important for large elements of global stiffness matrix than for user-defined or physical memory. The scheme will be carried out to divide blocks of the global stiffness matrix by user-defined sizes of arrays of the matrix. A typical procedure for this method is shown in Figure 6.11 with the maximum size of a block at 10.

block 1				block 2			block 3
1	3		9				8
	2	5	8				7
		4	7		6		6
			6	2	5		5
				1	4		4
					3	8	3
						7	2
							1

(a) symmetric blocked allocation.



(b) unsymmetric blocked allocation.

Figure 6.11. A typical block scheme of global stiffness matrix

Assemblage of symmetric or unsymmetric global matrix is as bellows. If the global matrix is symmetric, it comprised of the element stiffness matrix as in Figure 6.11(a). Then the blocked global matrix (STIFU) is assembled in vector form.
If the global matrix is unsymmetric, it must be shown as upper and lower parts. The assemblage in vector form is same as for the symmetric type, but the lower part of the unsymmetric global matrix is assigned as STIFL (Figure 6.11 (b)).

6.4.6 Assemblage of blocked global stiffness matrix.

After setting the size of a block and determining the equation numbers that belong to each block, element matrices must be added to each block. The algorithm is shown as Algorithm 1:

Blocked assemblage of symmetric and unsymmetric global stiffness matrix.

neql=1 (neql=number equations per block.)

mla =0 (mla=maximum number of diagonal element of blocked stiffness matrix.)

loop for blocks (nblock (number of block determined by sblock subroutine)

clear blocked global stiffness matrix. (if the matrix is symmetric, assemblage holds only the upper part of the matrix, and if unsymmetric , the assemblage process holds both upper and lower parts)

loop for element

add profile matrix (addban routine is used to add element matrix to the blocked symmetric or unsymmetric global matrix. When assembling the matrix, skip the row and column not included in this blocked equation number)

end of loop for element

neql=neql + ncolbv (ncolbv is equation number of this block)

mla=mal+ maxaj(neql)-1 (maxaj(neql) is maximum diagonal address of this block.)

end of loop for blocks

Algorithm 1. Block assemblage of element matrices

6.4.7 Fixed boundary condition procedure for blocked profile solver using separate routine.[33,48]

If the finite element problems have a fixed boundary condition, several kinds of method are needed to solve the condition. First, the traditional method is introduced, followed by the method of reducing load vector by blocked profile stiffness matrix.

6.4.7.1 Method of using a large number on the diagonal terms.

Matrix $[K]$ is first assembled without paying attention to the boundary conditions. Then, each specified value of the unknown $u_i = \bar{u}_i$ is introduced as follows:

- term k_{ii} is replaced by $k_{ii} + \alpha$ where there is a very large number with respect to all the other terms of k_{ij}; i.e., if α is chosen big enough, $k_{ii} + \alpha \cong \alpha$ (for example, in a six significant digit machine as $100.343 + 1.00000 \times 10^9 = 1.00000 \times 10^9$). The corresponding unknown x_i in

the same row would necessarily become negligibly small with respect to all the values of the other unknowns;

- all the terms of the right-hand side are replaced by their original values diminished by $f_j - k_{ji}\bar{u}_i = f_j^*$; the right-hand side of equation i is $-k_{ii}\bar{u}_i$; now comparing any equation before and after modification, the result is equation j (6.10):

- before $k_{j1}u_1 + k_{j2}u_2 + \ldots k_{ji}\bar{u}_i + \ldots k_{jn}u_n = f_j$

- after $k_{j1}u_1 + k_{j2}u_2 + \ldots k_{ji}x_i + \ldots k_{jn}u_n = f_j - k_{ji}\bar{u}_i$ (6.10)

Since $k_{ji}x_i$ is negligibly small with respect to all the other terms, the second equation can be identical to the unmodified equation with a proper choice of α, for equation i:

- before $k_{i1}u_1 + k_{i2}u_2 + \ldots k_{ii}\bar{u}_i + \ldots k_{in}u_n = f_i$

- after $k_{i1}u_1 + k_{i2}u_2 + \ldots \alpha x_i + \ldots k_{in}u_n = f_j - k_{ii}\bar{u}_i$ (6.11)

Subtracting the second from the modified equation results in:

$$k_{ii}\bar{u}_i - \alpha x_i = r_i + k_{ii}\bar{u}_i \tag{6.12}$$

Therefore,

$$r_i = -\alpha x_i \tag{6.13}$$

where r_i is the reaction corresponding to the specified value $\bar{u}_i$.

This method is very simple to code and gives, after solution, all the unknowns and reactions with no loss of accuracy. The matrix of coefficients retains its symmetry since only the diagonal term k_{ii} is modified.

6.4.7.2 Method of wiping rows and columns in place.

In this method, the load vector is modified as in the previous case for all specified values of the unknown $u_i = \bar{u}_i$ except for term f_i which is replaced by u_i:

$$\begin{aligned} f_j &= f_j - k_{ji}\bar{u}_i \quad \ldots\ldots\ldots\ldots j = 1,2,\ldots\ldots n(j \neq i) \\ f_i &= \bar{u}_i \end{aligned} \tag{6.14}$$

$$\begin{aligned} k_{ij} &= k_{ji} = 0 \quad \ldots\ldots\ldots j = 1,2,\ldots\ldots,n(i \neq j) \\ k_{ii} &= 1 \end{aligned} \tag{6.15}$$

Then, row and column intersections on i are wiped out and term k_{ii} is replaced by 1.

$$\begin{bmatrix} k_{11} & \cdots & k_{1,i-1} & 0 & k_{1,i+1} & \cdots & k_{1n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ k_{i-1,1} & \cdots & k_{i-1,i-1} & 0 & k_{i-1,i+1} & \cdots & k_{i-1,n} \\ 0 & \cdots & 0 & 1 & 0 & & 0 \\ k_{i+1,1} & \cdots & k_{i+1,i-1} & 0 & k_{i+1,i+1} & \cdots & k_{i+1,n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ k_{n1} & \cdots & k_{n,i-1} & 0 & k_{n,i+1} & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} u_1 \\ \vdots \\ u_{i-1} \\ u_i \\ u_{i+1} \\ \vdots \\ u_n \end{Bmatrix} = \begin{Bmatrix} f_1 - k_{11}\bar{u}_i \\ \vdots \\ f_{i-1} - k_{i-1}\bar{u}_i \\ \bar{u}_i \\ f_{i+1} - k_{i+1}\bar{u}_i \\ \vdots \\ f_n - k_{ni}\bar{u}_i \end{Bmatrix} \quad (6.16)$$

In this method, the reactions are not obtained directly.

6.4.8 Reactions.

For any equation i in which the unknown must take specified value as $\bar{u}_i$, the right-hand side member becomes an unknown quantity called a reaction. The method described in section 6.4.7.1 for application of the boundary conditions produced all the unknown values directly, as well as all the reactions. Another method, shown in section 6.4.7.2, requires a post-calculation form:

$$f_i = \sum_{j=1}^{n} k_{ij} u_j \quad (6.17)$$

6.4.9 Modify the load vector using blocked profile stiffness matrix for fixed boundary condition.

The traditional method for treating boundary conditions was described in the previous section. In this section, the load vector is modified using a blocked profile algorithm. This method is needed to assemble the

element matrix for the blocked global stiffness matrix including fixed boundary condition. This method is not only used to modify load vector, but also to calculate reaction and check for errors in the solution values.

The algorithm is the same as the method of wiping rows and column in place, except calculation of reaction and check of solution vector, and using the blocked stiffness matrix.

6.4.10 Algorithm of the proposed solution method.

General procedures of the finite element method for the proposed profile solution method will be described next.

1) Read information of nodal coordinates and degrees of freedom, element nodal connectivity, and material properties of elements.

2) Calculate equation numbers from nodal degrees of freedom. If fixed boundary conditions exist, the equation number is negative and it is necessary to use separate subroutine for fixed boundary conditions.

3) Estimate number of blocks and equation for each block of the global stiffness matrix by user-defined size of array or physical memory.

4) Build the element stiffness matrix.

5) Calculate equation numbers per element, the column height of each equation, and diagonal address of every equation.

6) Add element stiffness matrices to the blocked global stiffness matrix.

7) If fixed boundary conditions exist, use the boundary condition separate subroutine to revise the load vector.

8) After revising the load vector, decompose the blocked stiffness matrix, reduce load vector, and back substitute to get the solution vector. During these procedure, skip the negative equation number (related to the fixed boundary conditions).

9) Finally, reuse the subroutine related to the boundary condition to get reaction vector and check for errors in the solution.

6.4.10.1 Equation number of degrees of freedom per nodes.

The equation numbers of degrees of freedom per node are assigned as examples of Algorithm 2.

```
keq=0
do loop : ipoin=1,npoin (npoin=total number of node)
   do loop : idofn=1,ndofn(ndofn=nodal degrees of freedom)
      if ifpre(ipoin,idofn)=0 then keq=iabs(keq) + 1
        ifpre(ipoin,idofn)=keq
      else
if we do not need separate boundary condition then
  set     if ifpre(ipoin,idofn)=1 then ifpre(ipoin,idofn)=0
  skip next three lines.
       if ifpre(ipoin,idofn) greater then 0 then keq=iabs(keq) + 1
          ifpre(ipoin,idofn)=(-keq)
          else
         end if
       do loop end
    do loop end
```

Algorithm 2. Procedure of equation numbering.

6.4.10.2 Numbering of elements.

The numbering of element equations is determined by nodal connectivity and degrees of freedom, shown as Algorithm 3.

```
do loop element: 1,nelem(nelem=total number of elements)
    iev=1
    do loop node of element: 1,nnode(nnode=node per element)
      ipn=lnods(ielem,inode) (lnods=nodal connectivity of element)
      do loop of nodal degrees of freedom: 1,ndofn(ndofn=nodal degrees of greedom.)
        leqns(ielem,iev)= ifpre(ipn,idf) (leqns=element equation numbering matrix.)
        iev=iev+ 1
      end do loop of nodal degrees of freedom.
    end do loop of nodal connectivity
end do loop of element.
```

Algorithm 3. Build the element equation numbering.

6.4.10.3 Column heights and diagonal addresses.

The column height of every equation is the height from the diagonal element of global matrix to the envelope (skyline). The column height is a number from the diagonal element to the envelope except the diagonal one.

```
    do loop for element; 1,nelem(nelem=total number of element)
      do loop for element equation numbers; iev=1,nevab
        ia = leqns(ielem,ievab); (leqns=element equation connectivity matrix)
        If kboun equal to 1 then ia=iabs(ia); (kboun=fixed boundary condition parameter, if kboun=1,
  we consider fixed boundary condition, otherwise do not consider.)
     do loop for element equation numbers; jev=1,nevab
        ib = leqns(ielem,ievab)
        If kboun equal to 1 then ib=iabs(ib)
```

```
        If ib greater then 0
        jj=ib
        me=max0 (ii,jj)
        kht=mhigh(me)
        lht=iabs (ii-jj)
        mhigh(me) = max0 (kht, lht)
    end do loop for element equation numbers
end do loop for element
```

Algorithm 4. Column height of each equation.

```
    maxaj(1) = 1 ; (maxaj=diagonal address of global stiffness matrix)
    maxaj(2) = 2
    ma = 0
    do loop equations i=2,neqns (neqns=total number of equation)
        if mhigh(i) greater then ma, ma=mhigh(i); (mhigh(i)=column height of i equation)
        maxaj(i+ 1) = maxaj(i) +  mhigh(i) +  1
end do loop equations
neqn1=neqns+ 1
nwktl=maxaj(neqn1) - maxaj(1)
```

Algorithm 5. Diagonal addresses of equation.

6.4.11 Calculate blocks defined by user array size and equation number per block.

If the global stiffness matrix is larger than the computer memory, the stiffness must be divided into many blocks that do not exceed the memory, then stored on a hard disk. This section describes how to divide the global stiffness matrix and define equation numbers belonging to those blocks.

```
mtfu(used defined size of array for the global stiffness matrix.)
nwktl(total size of the global stiffness matrix.)
if nwktl less than
  nblock=1 (nblock=total number of blocks)
  ncolbv(1)=neqns (ncolbv=numbers of equation per block, neqns=total number of equations)
  icopl(1)=1(icopl=number of coupling block)
  istoh=nwktl(istoh=total numbers of equation this block)
if nwktl equal or greater than
  istorl=mtfu
  istoh=istorl
  do loop for equation number ; i=1,neqns(neqns=total number of equation)
    icl=maxaj(i+ 1)-maxaj(i)
    if icl greater then isoth
    write 'storage too smal'
  end do loop equation number
  nblock=0
  nn=0
  ib=0
```

```
  do loop equation number; i=2,neqns
140  continue
     ii = istoh - maxaj(i+ 1) + 1 +  nn
     if ii less then 0
       nn = maxaj(i) - 1
       nblock = nblock +  1
       ncolbv(nblock) = I - 1 - ib;(ncolbv=numbers of equation)
       ib = I - 1
       go to 140
     if ii equal or greater then 0 go to end do loop equation number
  end do loop equation number
  nblock = nblock +  1
  ncolbv(nblock) = neqns - ib
for establishing coupling blocks
  do loop for blocks; i=1,nblock
     icopl(i)=i
  end do loop for blocks
  nn = ncolbv(1)
  do loop for blocks; n=2,nblock
     iclm=0
     ncolb = ncolbv(n)
     do loop ncolb; i=1,ncolb (ncolb=number of equations n block)
       icl = maxaj(nn +  I + 1) - maxaj(nn+ i) - i - 1
       if icl greater then iclm
```

```
            iclm = icl
        end do loop ncolb
        j=n-1
150   if iclm less then 0   go to 180
          icopl(n) = j
          iclm = iclm - ncolbv(j)
          j=j-1
          go to 150
180         nn = nn +  ncolbv(n)
    end do loop nblock.
```

Algorithm 6. Build blocks and equation numbers for these blocks.

6.5 A blocked symmetric and unsymmetric linear equation solver with separated fixed boundary condition routine by profile method.

The main solution algorithm of the proposed solution method can now be written. This includes the possibility of treating the negative diagonal element of the global stiffness matrix, the blocked profile, and fixed boundary condition separately, as well as checking for errors in the solution vector. The procedure for writing the algorithm of the linear equation solution method is summarized as follows:

(1) To make the element matrix, use the same general procedure as for the finite element method.

(2) To make equation numbering of degrees of node requires a different approach. The equation numbering is positive as free degrees of freedom, but the negative numbering is concerned with fixed

(prescribed) boundary condition.

(3) To assemble the global stiffness matrix by element matrices held all equation, even if the negative equation number, it must add the element of matrix for every block.

(4) Before solving linear equations, if fixed (prescribed) boundary conditions exist, the load vector must be revised using SUBROUTINE DIRICH (Appendix 2). If the load is revised, IDX, IDY, and ISW arguments at SUBROUTINE DIRICH must set -1, 1, and -1.

(5) To solve linear equations, the element of blocked global stiffness matrix with negative equation numbering must be skipped. The solution procedure is the SUBROUTINE UNSOL (Appendix 3). When KBOUN at SUBROUTINE UNSOL is 1, and there is negative equation number, the solution procedure is: skip at equation numbering IFFIX argument concerning fixed boundary condition; if KKK is 1, decomposition of linear equation is held; and KKK is 2, reduction of load vector is calculated; and KKK is 3, backsubstitution is carried out to get result of displacements.

(6) To calculate reactions, again use the SUBROUTINE DIRICH but using solution vector instead of load vector. When calculating reaction, the arguments IDX, IDY, and ISW at SUBROUTINE DIRICH are set 0, -1, and 0.

(7) To check balance of solution vector, use SUBROUTINE DIRICH and set IDX=0, IDY=-1 ISW=0. The last solution vector provide values for displacements. The solution vector of negative equation number is the same as load vector and displacement vector has value of zero.

6.5.1 Step solution method for the embankment loading.

If the embankment is build on weak ground, the step solution is needed. All soil structure, including step embankment is auto meshed, and gravitational load of each step embankment is then calculated. The equation number for embankment and ground must be assigned for the

construction stage in the NEQLAY vector at the SUBROUTINE LINKIN. The NEQLAY vector contains information on all construction steps, which is needed to solve linear equation subroutine UNSOL and fixed boundary condition subroutine DIRICH. When calculating the load vector for fixed boundary condition and solving the linear equation, the layer information in the array NEQLAY is skipped and displacements are put zero value at every time step.

This proposed solution method is presented in Appendix 1, 2, and 3.

6.6 A finite element program structure of consolidation due to pumpage.

The fundamentals of finite element programming include the governing equations presented in section 3.4, the constitutive relationships and variable permeability of Chapter 4, and the numerical solution for the continuum theory of consolidation as described in Chapter 5. The next section will describe the program structure of the finite element model for consolidation due to pumpage.

6.6.1 Program structure of the continuum finite element due to pumpage.

The structure of the continuum model of the finite element due to pumpage is presented in flow chart 1.

Main program (Appendix 1):

The program defines the dimension of each matrix and vector, using subroutines GDATA, SET, ERROR2, LOADSM, LINKIN, BONCON, SBLOCK, GSTIGM, GLOAD, GASSEM, DIRICH, UNSOL, PRSOL, FRBAK, STRAIS, PRODSM, GRAPHI, and RESFLAVIA.

(1) Subroutine BONCON;
This reads boundary conditions and generates each nodal point.

(2) Subroutine FRBAK

This is used for back-substitution of the decomposed global stiffness matrix by subroutine PRSOL, using frontal solution method.

(3) Subroutine COISQX

This calculates $[L]$ matrix as COC and $[L]^T$ as COM at the equation (5.23).

(4) Subroutine DENSW

This calculates water density as a function of pore pressure and temperature.

(5) Subroutine DEP

This calculates stress-strain elasto-plastic $[D]$ matrix.

(6) Subroutine DESTRE

This reduces the stress vector to the yield surface for every yield criterion.

(7) Subroutine FCLOSE

This is used to close opened temporary units.

(8) Subroutine FDERIV

This calculates derivatives of the yield functions.

(9) Subroutine FL36QX

This calculates the element stiffness matrix for solid, pore water and thermal parts.

(10) Subroutine FLISQX

This calculates H and S matrix with equivalent load vectors QLOAD and SLOAD.

(11) Subroutine GDATA

This reads input data for calculation of the consolidation due to pumpage, such as nodal point coordinates, boundary condition, material properties, nodal connectivity and initial stresses.

(12) Subroutine GRAPHI.

This is needed to graph results, but is not used during analyses.

(13) Subroutine ILLCON
This calculates the largest number of stiffness matrix in the element matrix to multiply the number at ill conditioned element of global stiffness matrix related to the pore pressure and heat conduction term.

(14) Subroutine INSTRE
This calculates initial stresses at the Gauss point.

(15) Subroutine INTSEC
This evaluates stress variation from the elastic to plastic state at the second iteration.

(16) Subroutine INVAR
This calculates stress invariants for the elasto-plastic analysis, if needed.

(17) Subroutine JACOBB
This evaluates the Jacobian matrix, determinant and the inverse of Jacobian matrix.

(18) Subroutine GLOAD
This sets up equivalent nodal load vectors for each time step.

(19) Subroutine LOADSM
This makes smoothing matrix for temperatures and pore water pressures.

(20) Subroutine NODEXY
This generates interpolate coordinates of mid-side node for the 8-node quadrilateral element system.

(21) Subroutine FOPEN
This opens needed temporary files.

(22) Subroutine PCRITX
This expands the yield surface for elasto-plastic analysis.

(23) Subroutine PLISQX
This sets up the element stiffness matrix of solid part.

(24) Subroutine PRODSM
This is used to smooth initial temperatures and pore water pressures.

(25) Subroutine SET
This sets up the size of the global stiffness matrix.

(26) Subroutine BMATX

This sets up the $[B]$matrix.

(27) Subroutine DMATX

This makes $[D]$ matrix for plane strain, stress, or axis-symmetric case.

(28) Subroutine DBMATX

This produces $[B]$ and $[D]$.

(29) Subroutine DEPMAT

This calculates elasto-plastic $[D]$ matrix.

(30) Subroutine SFR2

This evaluates shape functions and their derivatives, also the upwind functions and their derivatives at the Gauss points.

(31) Subroutine FRSOL

This assembles the element matrix to the global one and triangular factorizes the global stiffness matrix by the frontal method.

(32) Subroutine GSTIFM

This sets up all of the element stiffness matrix and load vector.

(33) Subroutine STRAIS

This calculates stresses and strains from displacements.

(34) Subroutine STRLOA

This sets up equivalent nodal loads due to self-weight.

(35) Subroutine TLISQX

This sets up $[TLG][TR]$ and $[TP]$ matrix for the problem of thermo-coupling consolidation.

(36) Subroutine TLTS

This sets up $[TL]$ and $[TS]$ matrix for thermo-coupling consolidation problem.

(37) Subroutine TOISQX

This sets up $[TU]$ matrix for thermo-coupling problem.

(38) Subroutine GVELOC

This calculates nodal velocities and upwind coefficients alpha and beta.

(39) Subroutine XYZSTR

This calculates the global stresses after reducing the yield surface.

(40) Subroutine FYIELD

This calculates the value of the yield function at the current Gauss point.

(41) Subroutine ADDBAN

This sets up the element matrix to the global stiffness matrix by blocked assemblage with fixed boundary conditions, in which the element of fixed boundary part is added to the blocked global stiffness matrix. The method of adding global stiffness is a blocked profile type.

(42) Subroutine ADDRES

This calculates the diagonal addresses from calculated column height of the global matrix.

(43) Subroutine GASSEM

This assembles the element stiffness matrix to the global one (subroutine ADDBAN) and makes a blocked global stiffness matrix of the profile type.

(44) Subroutine DIRICH

This treats the fixed boundary condition part. It is used before solving the linear equation to revise the load vector with fixed boundary condition. And it will be used to calculate reactions,

etc. using results from solved linear equations.

(45) Subroutine COLMHT
This calculates column heights of the blocked global stiffness matrix, even if fixed boundary conditions occur.

(46) Subroutine LINKIN
This makes equation numbers of nodal degrees of freedom, connectivity arrays, column heights, and diagonal addresses.

(47) Subroutine SBLOCK
This forms blocks to store segmented global stiffness matrix.

(48) Subroutine UNSOL
This solves symmetric and unsymmetric finite element static equilibrium equations out-of-core, using a blocked profile scheme with fixed boundary conditions and layered constructions.

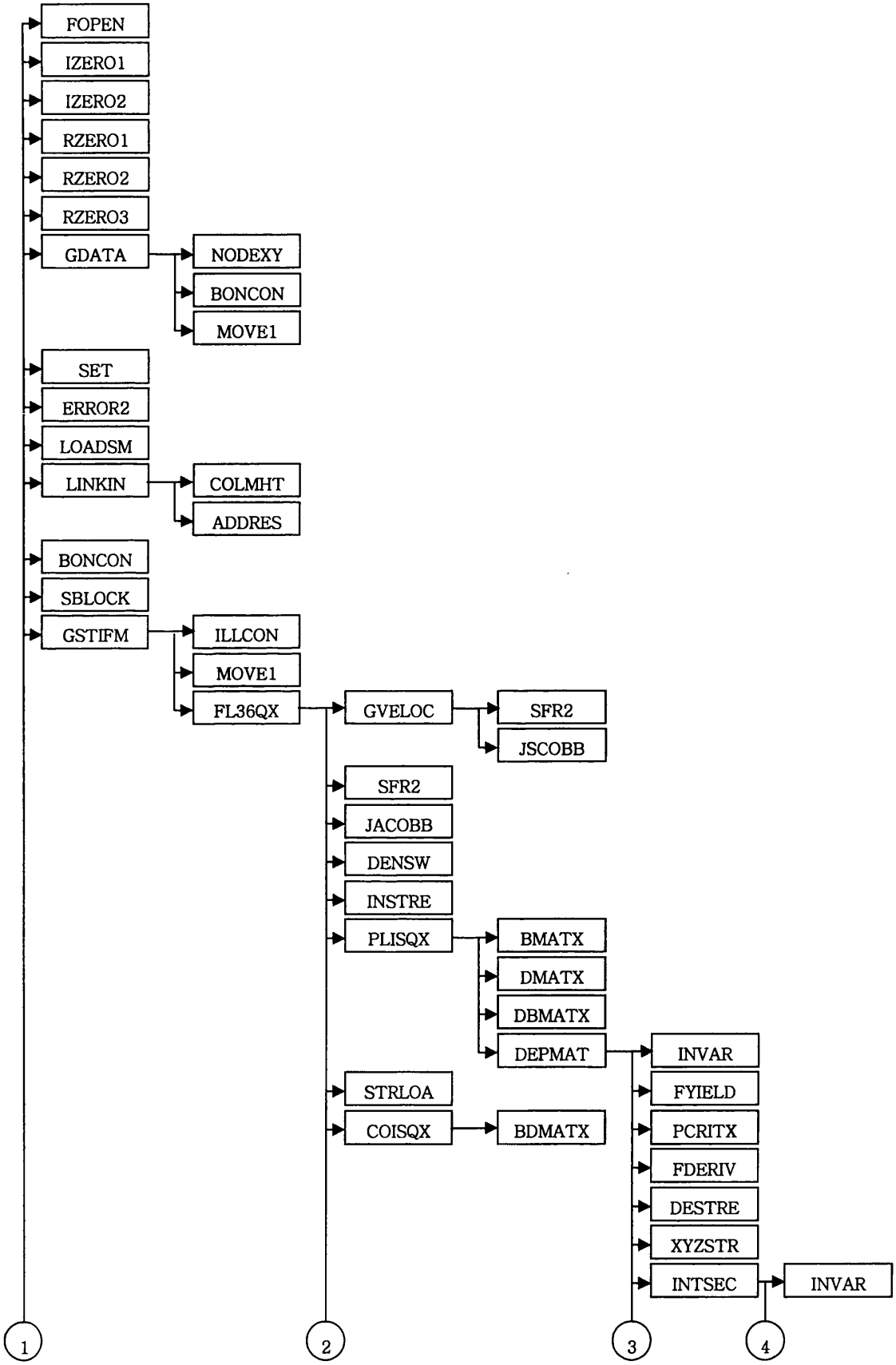
(49) Subroutine RESFLAVIA
This changes evaluated displacements, stresses and strains to the data for post-processing program GID.

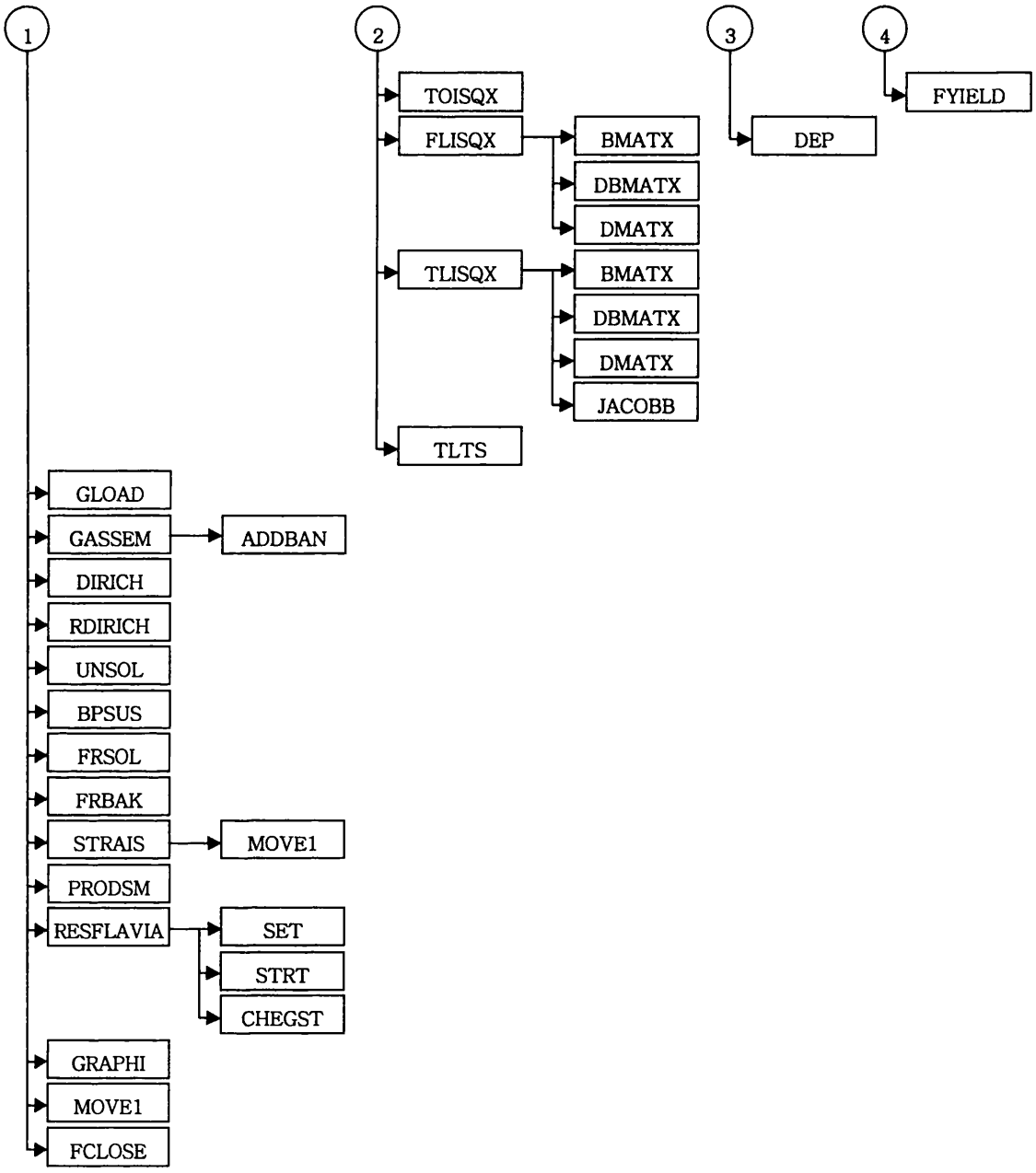
6.7 Concluding remarks

This chapter provides all algorithms of a finite element continuum model for consolidation due to pumpage including pre- and post-processing procedures. Flow chart 1 shows the organization of the developed program.

Flow chart 1.

Flow chart of the program for consolidation due to pumpage


FOPEN
IZERO1
IZERO2
RZERO1
RZERO2
RZERO3
GDATA
NODEXY
BONCON
MOVE1
SET
ERROR2
LOADSM
LINKIN
COLMHT
ADDRES
BONCON
SBLOCK
GSTIFM
ILLCON
MOVE1
FL36QX
GVELOC
SFR2
JSCOBB
SFR2
JACOBB
DENSW
INSTRE
PLISQX
BMATX
DMATX
DBMATX
DEPMAT
INVAR
FYIELD
PCRITX
FDERIV
DESTRE
XYZSTR
INTSEC
INVAR
STRLOA
COISQX
BDMATX
1
2
3
4


1
2
3
4
TOISQX
FLISQX
BMATX
DBMATX
DMATX
TLISQX
BMATX
DBMATX
DMATX
JACOBB
TLTS
DEP
FYIELD
GLOAD
GASSEM
ADDBAN
DIRICH
RDIRICH
UNSOL
BPSUS
FRSOL
FRBAK
STRAIS
MOVE1
PRODSM
RESFLAVIA
SET
STRT
CHEGST
GRAPHI
MOVE1
FCLOSE

Chapter 7

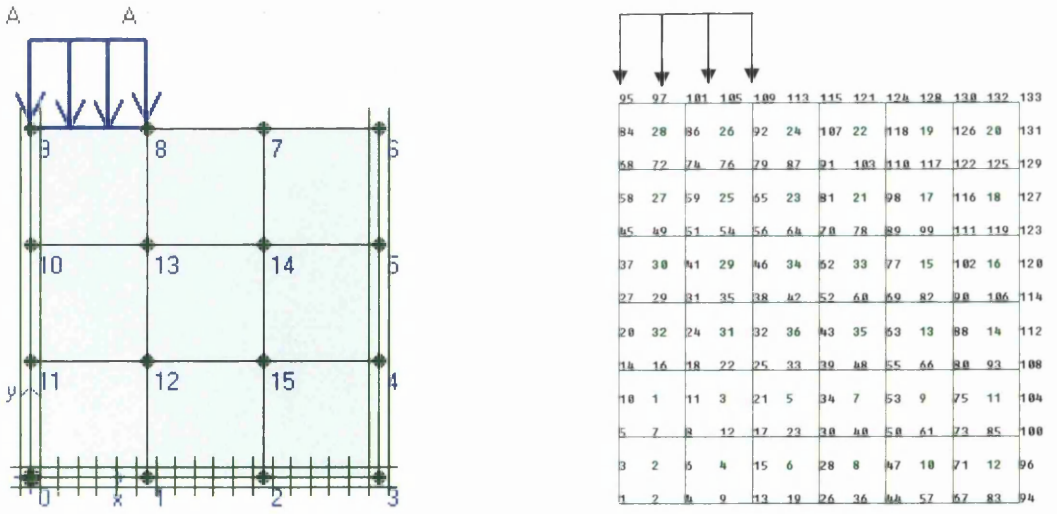
Verification of the developed program

The program used in this thesis is verified in this chapter through comparison with the PLAXIS[66] program, which is used by geotchnicians around the world to modell and solve consolidation and geotechnical problems,

A simple problem was tested because of difference between the two programs in their analytical approach to consolidation problem. The program developed in this thesis used quadrilateral elements instead of the triangular elements of the PLAXIS program, so the example model is differ.

7.1 Example modelling of a simple consolidation problem.

Modelling of a simple consolidation problem is shown as Figure 7 (a) for the PLAXIS program and Figure 7 (b) for the developed program of this thesis.

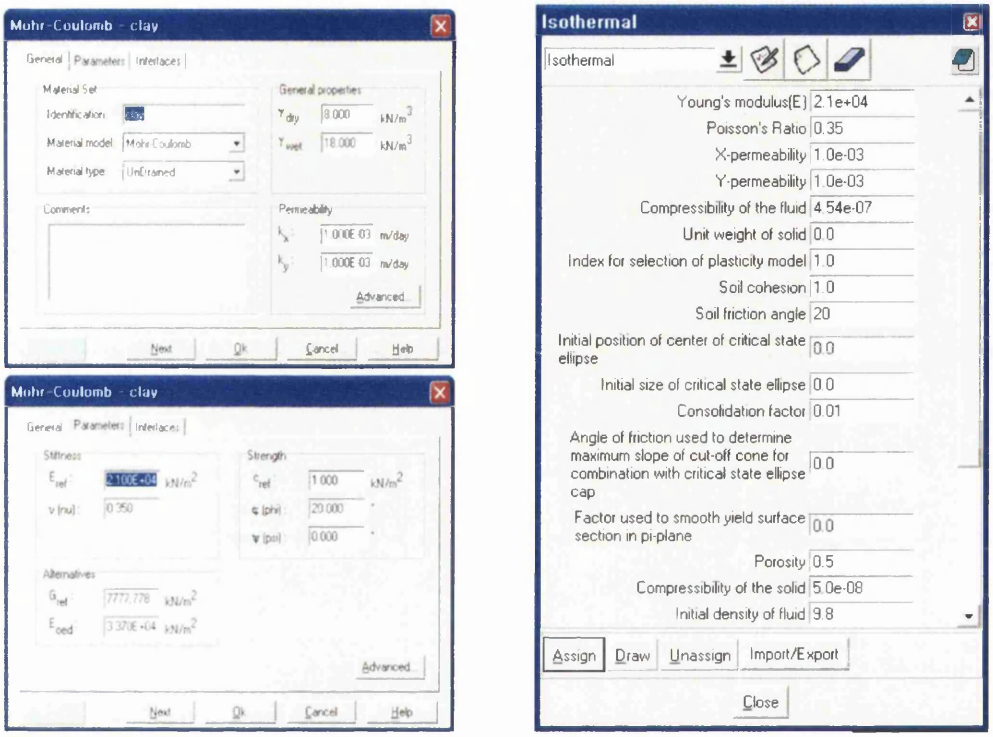


(a)The PLAXIS program modelling. (b) The developed program modelling.

Figure 7.1 Example modelling of a consolidation problem

7.2 Material properties and other conditions.

Material properties for the example in Figure 7.1 with a uniform load of 9.8 Kn/m as follows:

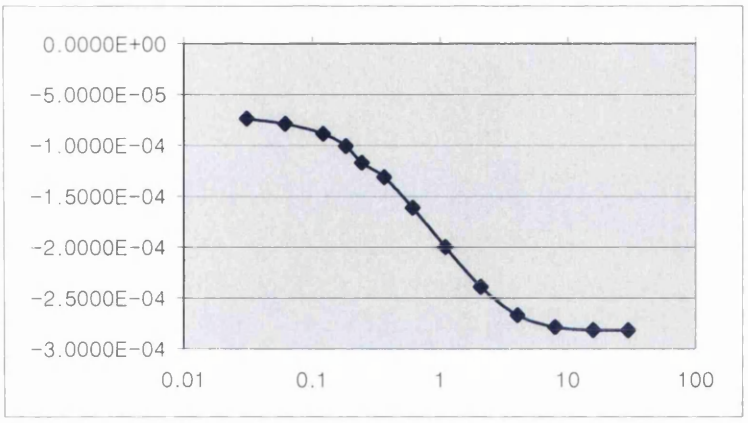


(a) PLAXIS data (b) Developed program data

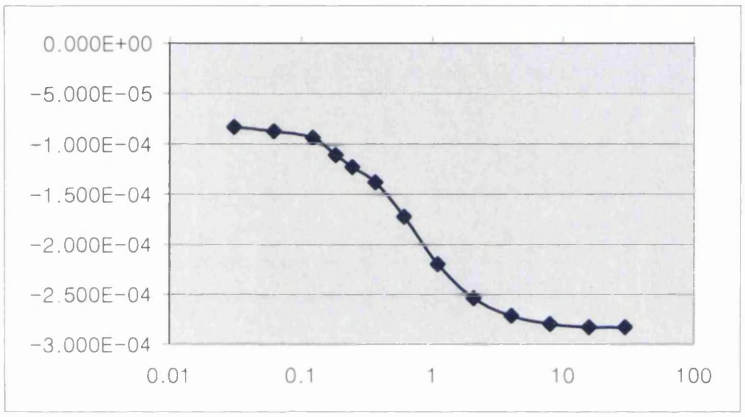
Figure 7.2 Material properties for the example problem

7.3 Comparison of the results.

The results for the example problem at the same point is as similar, but not an exact match. Some differences because the algorithm of the developed program is not the same as that used in the PLAXIS program. This is illustrated in Figure 7.3.



(a) PLAXIS results.



(b) The developed program results.

Figure 7.3 Results of Y-displacement at the point (X=1.0, Y=2.0)

7.4 concluding remarks.

The similarity in the results obtained by each program validates the accuracy of the developed program for solving.

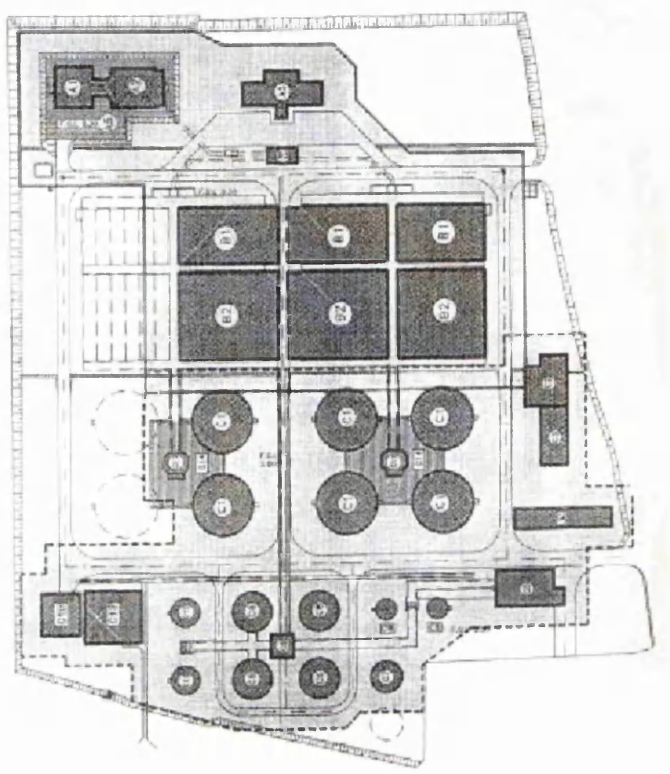
Chapter 8

A practical application of pumpage consolidation at Kimhae, South Korea[47]

The Kimhae sewage treatment site is located in the plain of Kimhae, west of Pusan city, South Korea, along the banks of the Nak Dong River. The facility encompasses a surface area of some 160000m^2, divided into two phases, each 80000m^2. It is located on a former rice field over highly compressible clay layers with depths varying between 25 and 43 m as a result of marine deposit in this 20 km×20 km flood plain.

The promotion of consolidation by pumpage using the Menard Vacuum Consolidation method was developed, for the first time in South Korea, for the construction of this sewage treatment plant. The extreme soil conditions, structure loads, and specifications of this project led to new calculation methods at the design and monitoring stages. The project started in January 1995 and the plant began operating in January 2000, upon completion of the first phase. The general layout of the plant is shown in Figure 8.1.

Structural loads ranged from 3.3 to 15.5 t/m^2 (Table 8.1) and foundation depth ranged from 0 to $-7\,m$ from the final ground elevation. The sewage treatment plant was designed for a gravity process, which excluded pumping of sewage waters inside the plant. Sewage starts from the upper Screen House at an elevation of $+5.3\,m$, and water is conveyed into the river at an elevation of $+1.8\,m$.



ZONE	Design Load
A1	q=7.00(T/M2)
A2	q=10.44(T/M2)
A3	q=5.80(T/M2)
B1	q=6.65(T/M2)
B2	q=6.70(T/M2)
B3	q=9.20(T/M2)
C1	q=8.86(T/M2)
C2	q=6.60(T/M2)
C3	q=3.30(T/M2)
C4	q=4.45(T/M2)
C5	q=6.60(T/M2)
C6	q=15.51.3(T/M2)
C7	q=6.44(T/M2)
C8	q=6.67(T/M2)
C9	q=5.77(T/M2)
C10	q=5.35(T/M2)
C11	q=8.86(T/M2)
C12	q=6.60(T/M2)
C13	q=5.00(T/M2)

Figure 8.1 Plan of Kimhae sewage treatment plant

8.1 Characteristics of the subsoil and typical cross-section.

Preliminary soil investigation consisted of standard penetration, cone penetration, and pressure meter and laboratory tests. The project included 24 quality control points located under the proposed plant facility. The subsoil characteristics are not uniform and three different soil layers can be distinguished with a compressible soil thickness ranging from 25 to 43 m.

- Upper layer (silt with sand) : $h = 4$ to 7 m
- Intermediate layer (compressible saturated clay) : $h = 20$ to 35 m
- Bedrock (weathered rock) : $h = 25$ to 43 m

A more detailed description of the subsoil conditions, in the form of a typical cross-section with the recorded N-STP values for several representative bore holes, is shown in Figure 8.2.

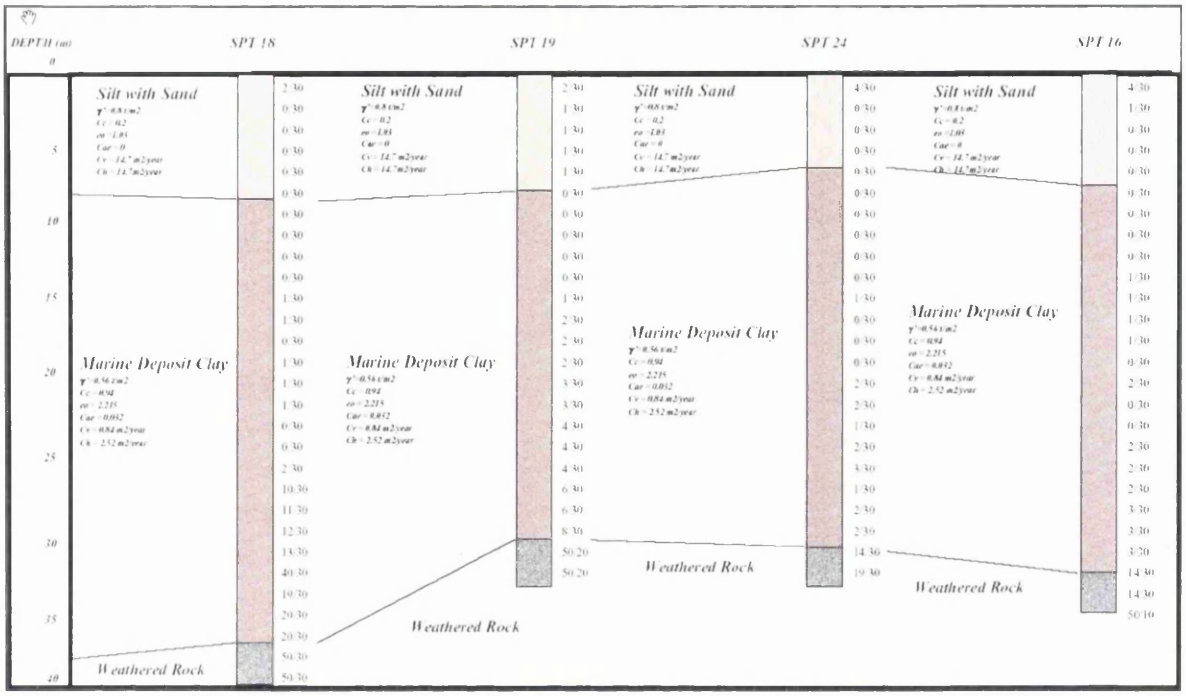


Figure 8.2 Typical cross-section with phase1 / STP tests

8.2 Results of field consolidation by pumpage at Kimhae.

The recorded settlement ranged from 314 cm to 548 cm for surcharge height varying from 6.5 to 15.5 m during an average nine months of pumpage. The typical settlement, illustrated by Figure 8.3, shows the efficiency of the Menard method, with an acceleration of the settlement by two or three times indicated by the steep increase in the slope at vacuum start. In terms of calibration coefficient β, the average value found on 24 settlement plates was $\beta = 0.91$.

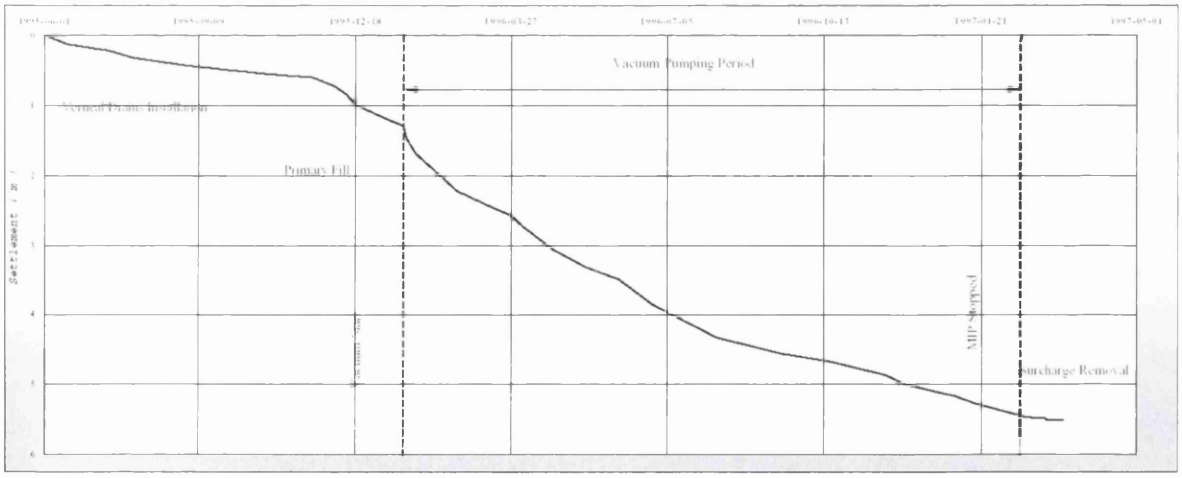


Figure 8.3 The typical settlement curve at Kimhae

As soon as the vacuum was turned off and additional surcharge was removed, settlement stopped and a slight rebound was recorded, as shown in Figure 8.4. Settlements stabilised after two weeks, and residual settlements were not observed.

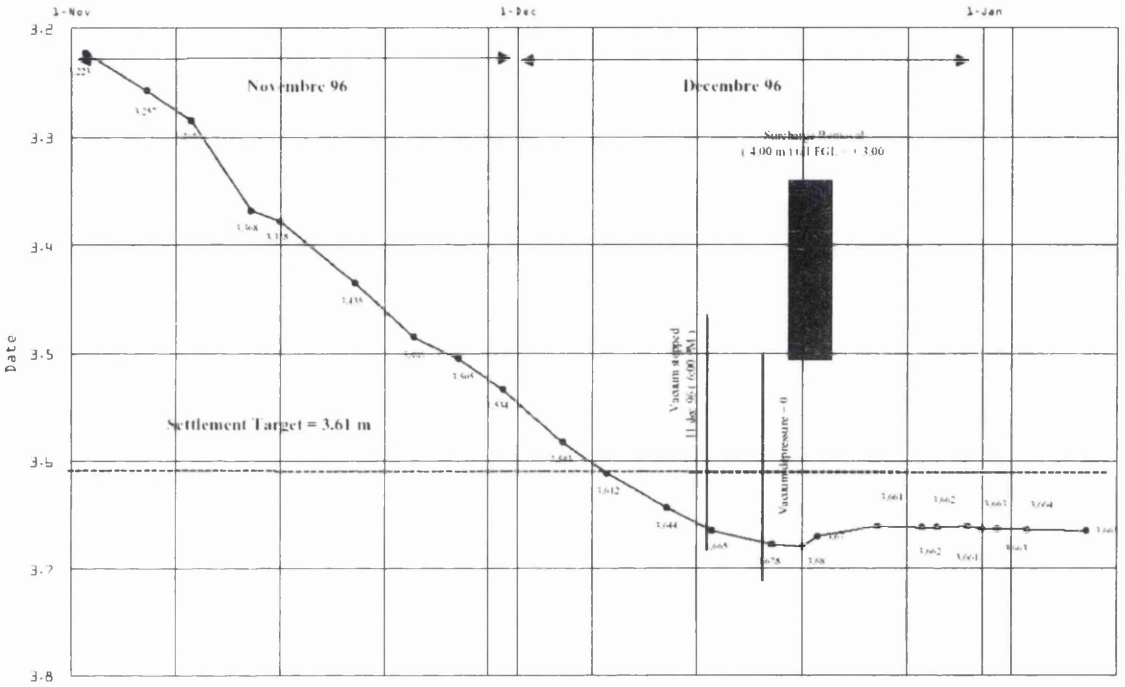


Figure 8.4 Rebound curve after vacuum finished

8.3 Concluding remarks.

The Kimhae sewage treatment project, as reported by Ihm and Masse[49], successfully tested the practical application of vacuum consolidation for soil improvement under a concrete structure with severe settlement criteria. This project shows the effectiveness, both technically and economically, of vacuum-assisted consolidation as an alternative to piles and /or conventional pre-loading.

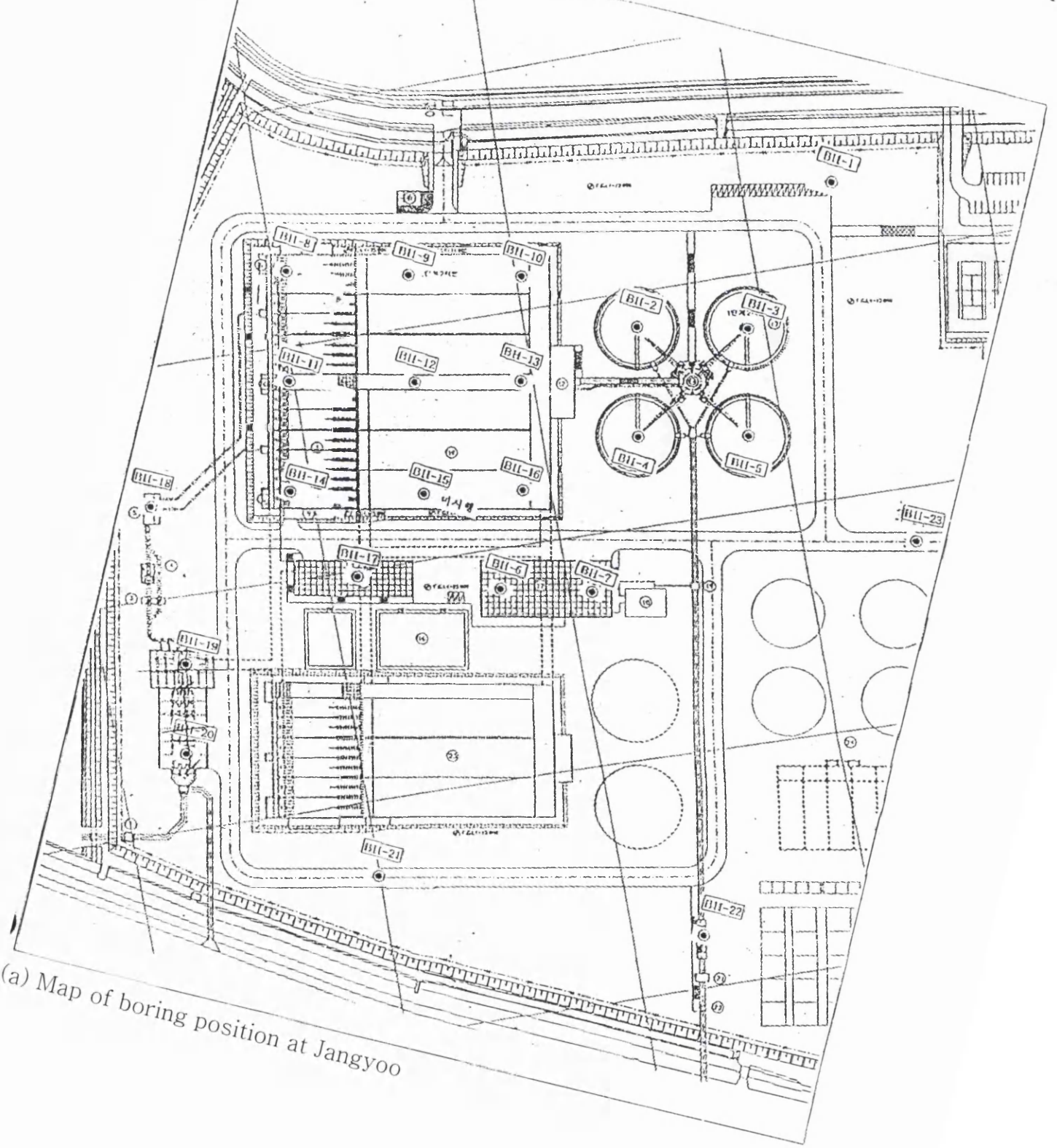
Chapter 9

Numerical results and verification for consolidation due to pumpage.

A numerical example of the proposed solution method for consolidation due to pumpage and the field measurements at Jangyoo (South Korea) are presented in this chapter. The modelled results of the continuum finite element method with pumpage are compared with field measurements for the purpose of verification, showing the utility of finite modelling with linear equation for the treatment of separated fixed boundary condition.

9.1 Plan of Menard Vacuum Consolidation at Jangyoo by pumpage. [15,26,65,74,75,76]

A modelling of the Menard vacuum system for pumping the sewage plant at Jangyoo is described in this section. The position of bore holes and profile of the sub-soil are given in Figure 9.1.



(a) Map of boring position at Jangyoo

Area	*Design Load*	*Foundation Depth from FGL*	*FGL*
AS1	8.29 t/m2	3.5 m	+3.00 m
AS2	4.37 t/m2	3.65 m	+3.00 m
AS3	18 t/m2	12.5 m	+3.00 m
AS4	10.6 t/m2	8 m	+3.00 m
AS5	road		+3.00 m

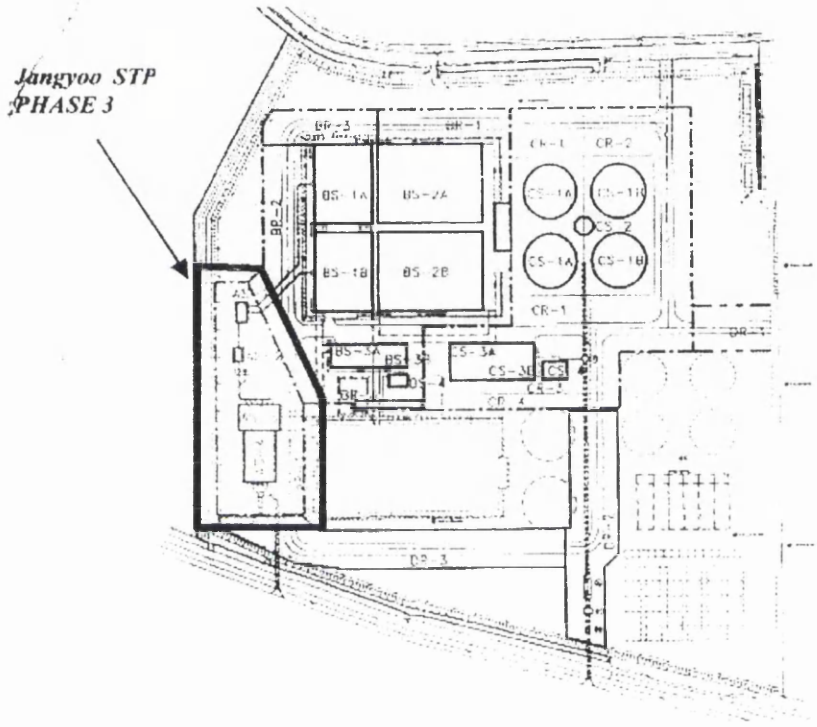
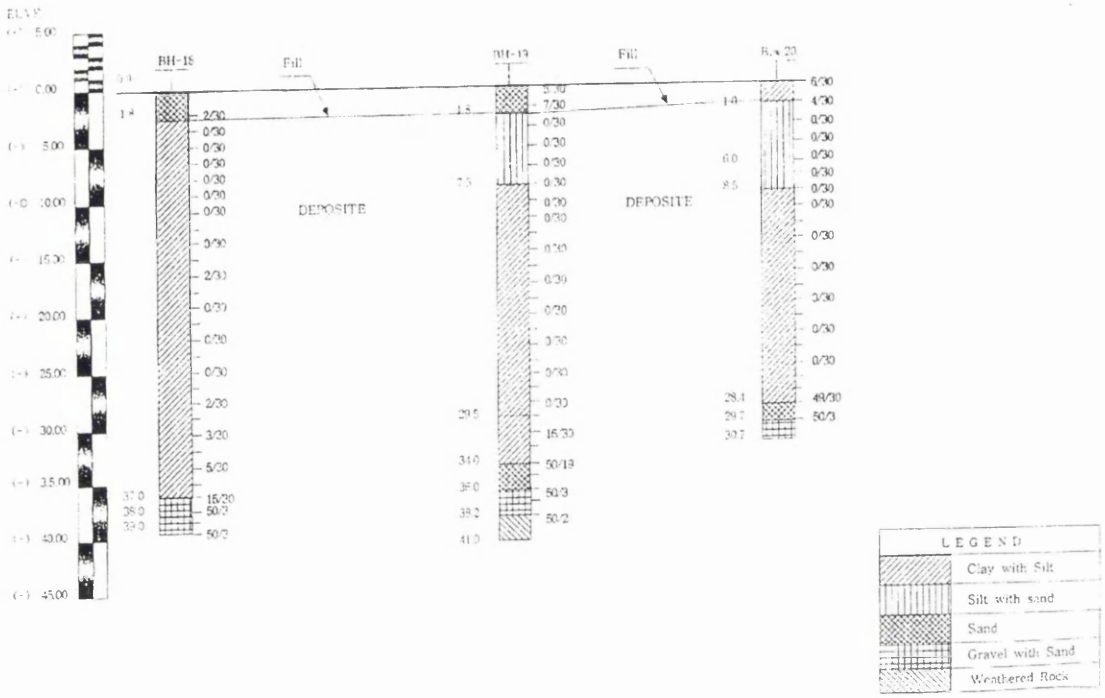


Fig 1 : General Layout plan

(b) General layout of Jangyoo plant plan



- upper layer : Clayey Silt: h = 15 m

γ'	0.58 t/m2	Cae second. consolidation	0.0364
Cc Compression index	0.91	Cv (coef of vertical consolidation)	1.32 m2/year
eo Void ratio	2.012	Ch (coef of horizontal consolidatiton)	2.64 m2/year

- Medium : Compressible saturated Clay : h = 9 to 16 m

γ'	0.58 t/m2	Cae second. consolidation	0.0484
Cc Compression index	1.21	Cv (coef of vertical consolidation)	1.32 m2/year
eo Void ratio	2.012	Ch (coef of horizontal consolidatiton)	2.64 m2/year

- Bottom layer (above bedrock) : Stiffer Clay h=4m

γ'	0.58 t/m2	Cae second. consolidation	0.008
Cc Compression index	0.20	Cv (coef of vertical consolidation)	1.32 m2/year
eo Void ratio	0.98	Ch (coef of horizontal consolidatiton)	2.64 m2/year

(c) Profile of sub-soil and properties of the soil

Figure 9.1 Map of the plan and profile of the sub-soil

The plan of embankments and some material properties used to calculate settlement by Barron's vertical drain are shown in Figure 9.2. Properties of sub-soil are also used in the finite element simulation.

Meshing sizes in the finite element simulation are the same as the distance between the installed Menard tubes. This is required to analyse the fixed boundary condition of pore pressures accurately, instead of relying on changing material properties and other meshing sizes decided by the user. Material properties are determined by laboratory experiments and field tests such as STP (Table 9.1). The typical modelling of the Jangyoo sewage plant by the finite element method is shown in Figure 9.2.

Next, the plan of embankment, start and stop of vacuum pumpage, and measured settlements are shown (Figure 9.2). Five embankment layers promote consolidation settlement. The Menard vacuum tube was installed before the construction of embankment, and pumpage held after embankment. Once the desired settlement was achieved, pumpage was stopped. Field measurements of settlement were performed over a period of nine months of pumpage, as well as after pumpage had been stopped. These field measurements are needed to determine the final settlement.

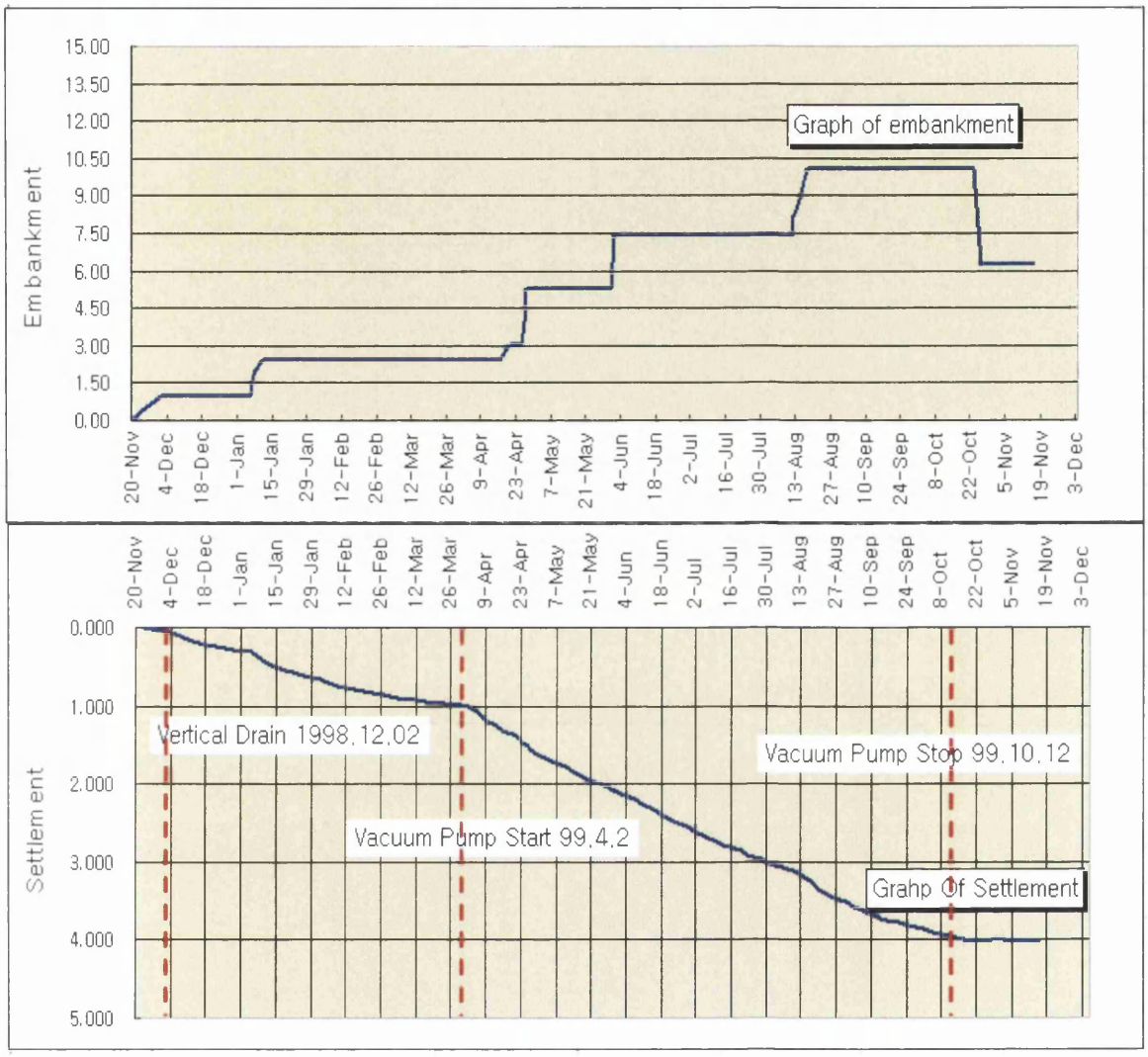
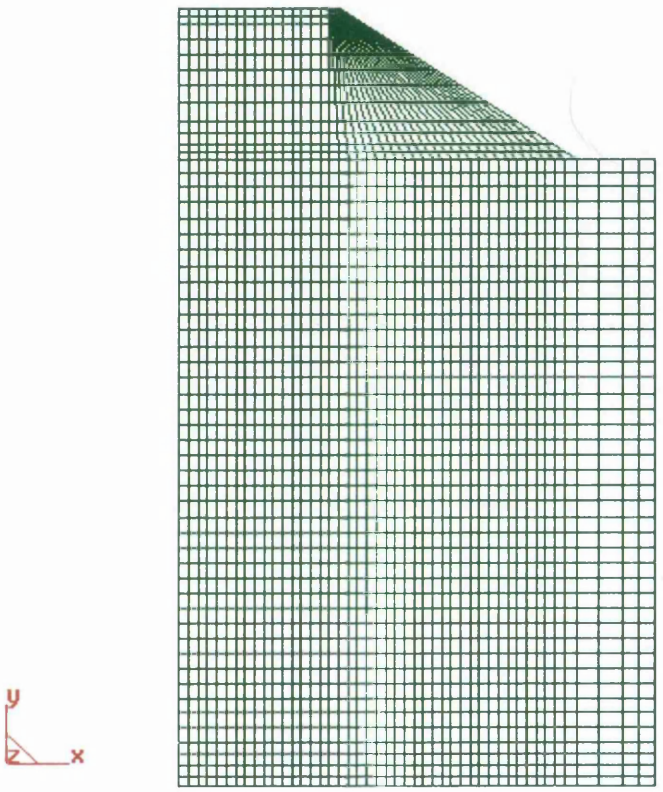


Figure 9.2 Plan of embankment, start and stop of vacuum pumpage, and measured settlements

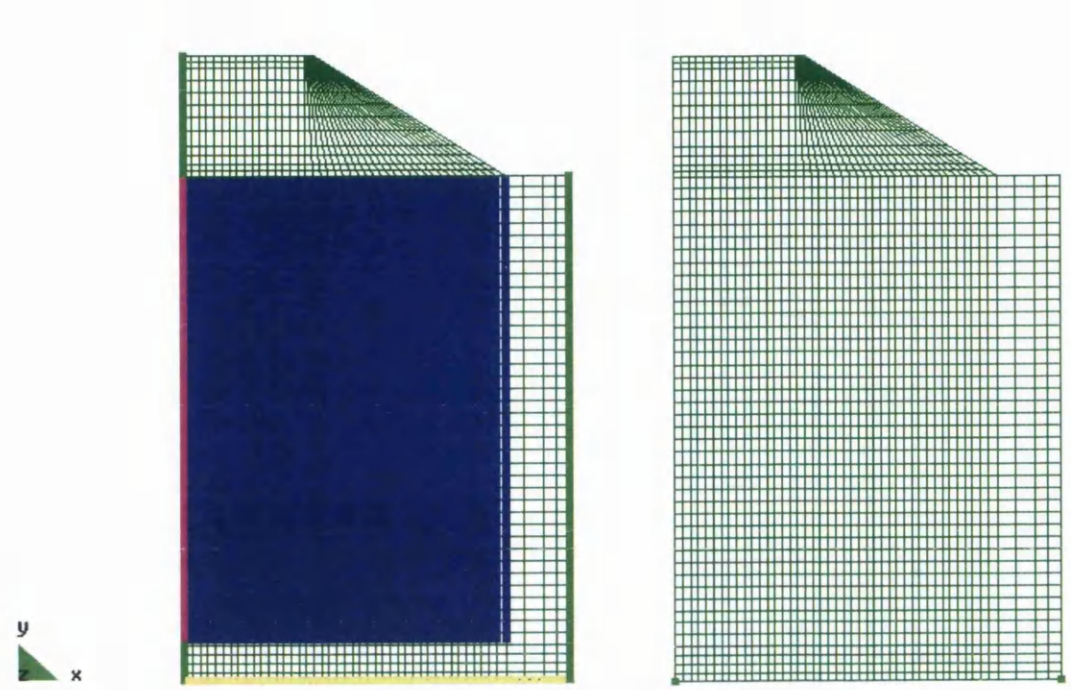
9.2 A continuum finite element modelling of consolidation at Jangyoo by pumpage.

Finite element simulation is needed to solve the governing differential equation numerically, thus giving accurate results for natural facts.
In this section, a numerical model by finite element method is shown to solve the practical promotion of consolidation at Jangyoo.
The narrow part of Figure 9.3 (a) represents the installed Menard vacuum tube with the width of meshes beign the same as field holes for

convenience; the wide parts of mesh are user defined for sub-ground; and upper parts are embankments.



(a) Meshes of finite element modelling at Jangyoo



(b) Boundary conditions for the modelling

(c) Material properties of the model

Figure 9.3 Example of finite element modelling by pumpage at Jangyoo

Boundary conditions shows in the Figure 9.3 (b), left- and right-end

boundary conditions fixed as 0.0 at x-directions. Boundary conditions of bottom are fixed at y-direction, and pore-pressure. The last boundary conditions have fixed value as -70.0 Pascal. This is needed to calculate settlement due to pumpage. The named interior boundary condition is illustrated in blue.

Figure 9.3 (c) shows kinds of material properties for the sub-soil and embankments. Nodal points at the left- and right-end sides have x-directional fixed displacement boundaries and pore pressures are fixed by zero; and there is another bottom of y-direction, all points fixed as zeros for all displacements and pore pressures.

9.3 Material properties and other analyses conditions.

Because soils are far more complex materials than concrete or steel, exact results cannot be obtained by experimental data. It is very important to calculate precise properties and empirical data in the finite element simulation. This is achieved through the numerical method that has been developed in this thesis. The typical properties of soil, using numerical analysis, are given in Table 9.1.

N	1	2	3	4	5
E (kN/m^2)	5.07×10^{9}	2.0×10^{7}	9.7×10^{4}	8.5×10^{3}	2.3×10^{3}
ν	0.25	0.3	0.32	0.32	0.32
$K_x(m/day)/\gamma_w(kN/m^3)$	0.0	8.64×10^{-11}	3.73×10^{-5}	2.64×10^{-3}	2.78×10^{-3}
$K_y(m/day)/\gamma_w(kN/m^3)$	0.0	8.64×10^{-11}	3.73×10^{-5}	2.64×10^{-3}	2.78×10^{-3}
$1/K_w(kN/m^2)$	4.54×10^{-7}	4.54×10^{-7}	4.54×10^{-7}	4.54×10^{-7}	4.54×10^{-7}
$\gamma_s(kN/m^3)$	28.59	23.67	5.68	5.68	5.68
MYIELD	1	1	1	1	1
+ $c(kN/m^2)$	500	0	50.25	40.40	50.25
ϕ( )	70	35	0	25.8	26.5
p_{c0}	0.0	0.0	0.0	0.0	0.0
P_{r0}	0.0	0.0	0.0	0.0	0.0
XI	0.0	0.0	15.21	17.21	7.5
ϕ_{c0}( )	0.0	0.0	0.0	0.0	0.0
$\pi-plane$, k	0.0	0.0	0.0	0.0	0.0
Porosity n	0.0	0.1	0.49	0.66	0.66
$1/K_s$ (kN/m^2)	5.0×10^{-8}	5.0×10^{-6}	5.0×10^{-4}	5.0×10^{-4}	5.0×10^{-4}

$\rho_w(kN/m^3)$	9.8	9.8	9.8	9.8	9.8
N	6	7	8	9	10
E (kN/m^2)	$2.3x10^4$	$2.3x10^5$	$2.3x10^5$	$2.3x10^5$	$2.3x10^5$
ν	0.3	0.3	0.3	0.3	0.3
$K_x(m/day)/\gamma_w(kN/m^3)$	$8.5x10^{-2}$	0.0	0.0	0.0	0.0
$K_y(m/day)/\gamma_w(kN/m^3)$	$8.5x10^{-2}$	0.0	0.0	0.0	0.0
$1/K_w(kN/m^2)$	$4.54x10^{-7}$	$4.54x10^{-7}$	$4.54x10^{-7}$	$4.54x10^{-7}$	$4.54x10^{-7}$
$\gamma_s(kN/m^3)$	17.74	17.74	17.74	17.74	17.74
MYIELD	1	1	1	1	1
$c(kN/m^2)$	10.4	10.4	10.4	10.4	10.4
ϕ( )	28	23	23	23	23
p_{c0}	0.0	0.0	0.0	0.0	0.0
P_{r0}	0.0	0.0	0.0	0.0	0.0
XI	0.0	0.0	0.0	0.0	0.0
ϕ_{c0}( )	0.0	0.0	0.0	0.0	0.0
$\pi - plane$, k	0.0	0.0	0.0	0.0	0.0
Porosity n	0.34	0.3	0.3	0.3	0.3
$1/K_s\ (kN/m^2)$	$5.0x10^{-6}$	$5.0x10^{-6}$	$5.0x10^{-6}$	$5.0x10^{-6}$	$5.0x10^{-6}$
$\rho_w(kN/m^3)$	9.8	9.8	9.8	9.8	9.8

N	11	12	13
E (kN/m^2)	$2.3x10^5$	$2.3x10^5$	$2.3x10^5$
ν	0.3	0.3	0.3
$K_x(m/day)/\gamma_w(kN/m^3)$	0.0	0.0	$8.5x10^{-12}$
$K_y(m/day)/\gamma_w(kN/m^3)$	0.0	0.0	$8.5x10^{-12}$
$1/K_w(kN/m^2)$	$4.54x10^{-7}$	$4.54x10^{-7}$	$4.54x10^{-7}$
$\gamma_s(kN/m^3)$	17.74	17.74	17.74
MYIELD	1	1	1
$c(kN/m^2)$	10.4	10.4	50.4
ϕ( )	23	23	23
p_{c0}	0.0	0.0	0.0
P_{r0}	0.0	0.0	0.0
XI	0.0	0.0	0.0
ϕ_{c0}( )	0.0	0.0	0.0
$\pi - plane$, k	0.0	0.0	0.0
Porosity n	0.3	0.3	0.34
$1/K_s\ (kN/m^2)$	$5.0x10^{-6}$	$5.0x10^{-6}$	$5.0x10^{-6}$
$\rho_w(kN/m^3)$	9.8	9.8	9.8

Table 9.1 Typical material data

Another condition of simulation occur when embankments start and are removed and pumpage begins and stops. The procedures and time steps are fitted as actual construction steps as in Figure 9.2. Permeability constants are m/day in the modelling.

9.4 Results of the continuum modelling due to pumpage and comparison with field measurements.

Two programs were used for analysing the continuum consolidation model due to pumpage. A revised PLASCON[52] was employed for the purpose of evaluating the problem of multi-phase consolidation settlement. The other key program is GID,[15] which has been developed as a commercial pre- and post-processing program for the finite element method for consolidation due to pumpage.

9.4.1 Vertical settlements.

The largest vertical settlement figure of the last time step is given in Figure 9.4. The curve of settlement is similar to the field measurement, showing that this simulation method is useful to predict practical problems by the FEM.

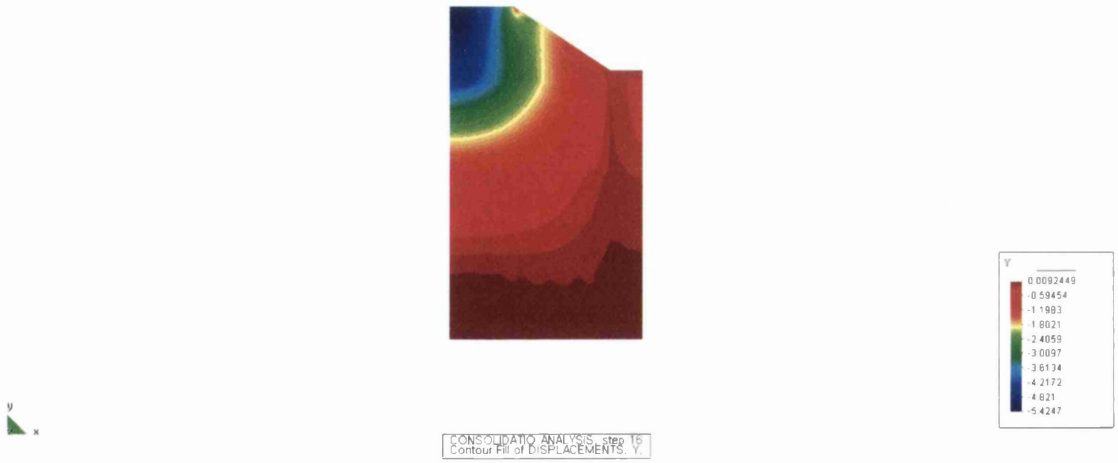


Figure 9.4 Vertical settlements

9.4.2 Pore pressures.

Diagram of typical pore pressures at the last time step is shown in Figure 9.5. The results are almost exact when compared with the field measurement.

Figure 9.5 Pore pressures

9.4.3 Horizontal displacement.

At times, it is necessary to see horizontal movement of the sub-ground, so a horizontal displacement graph of the last time step is given in Figure 9.6. These values could not be compared with field values because that there is no filed measurement data for horizontal displacement.

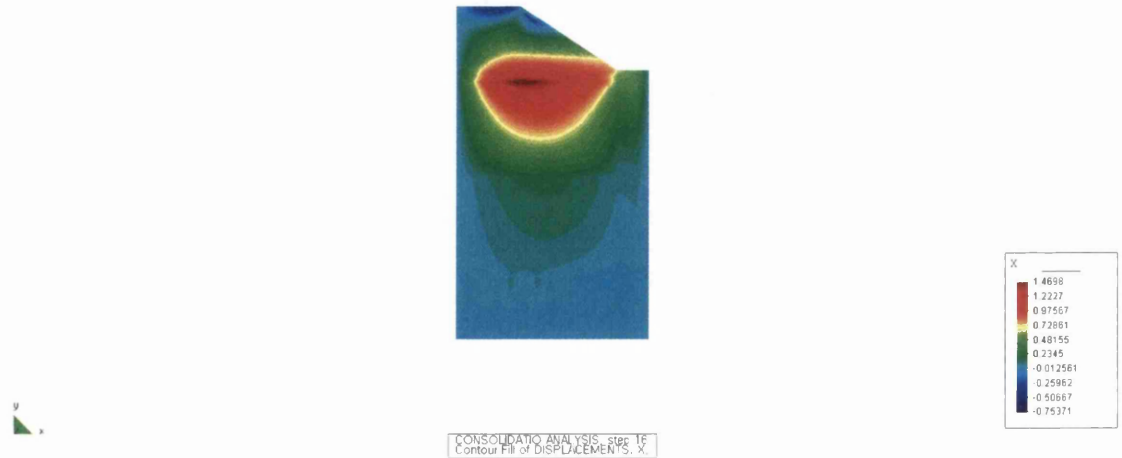
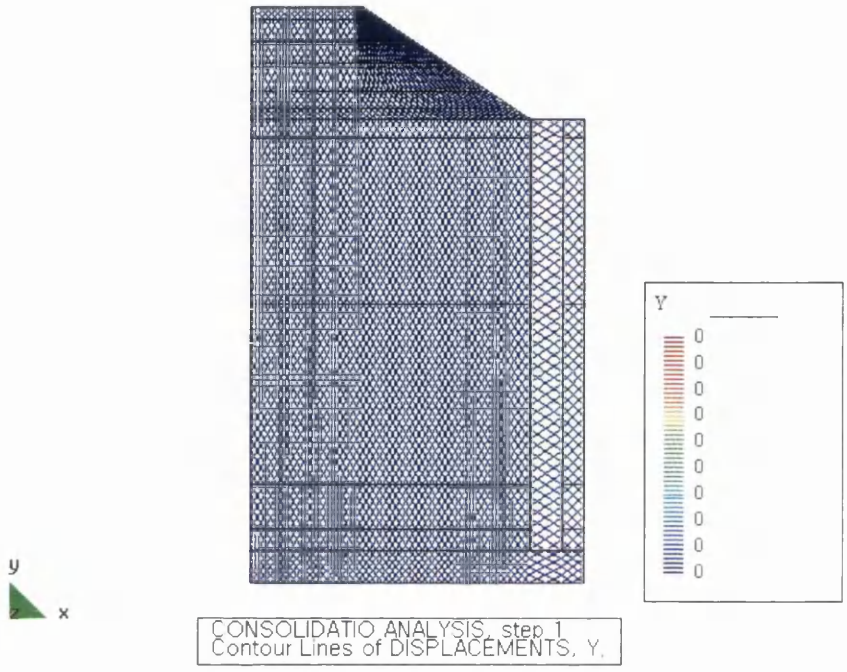
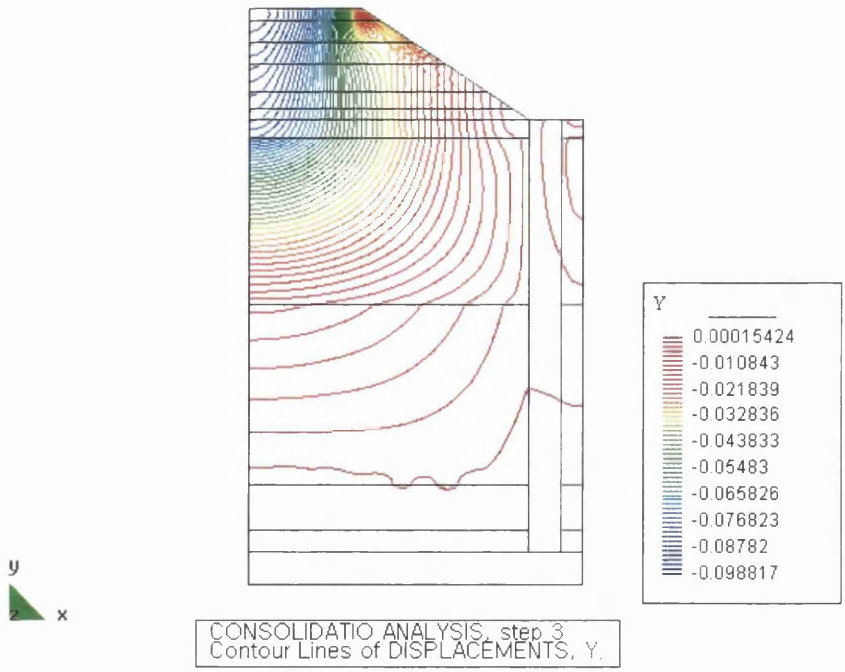


Figure 9.6 Horizontal displacement

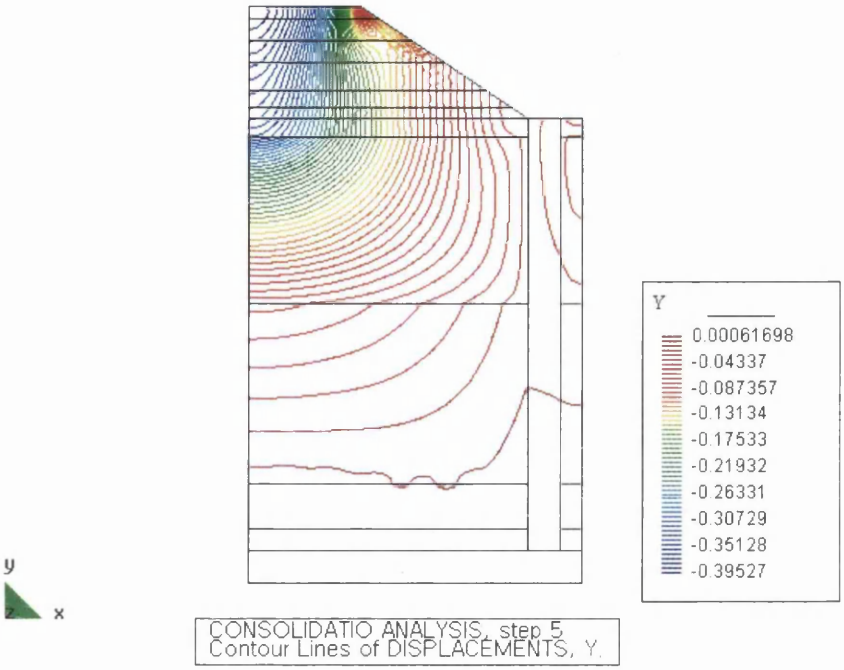
9.4.4 Contour of y-displacements plots for some time and embankment steps.



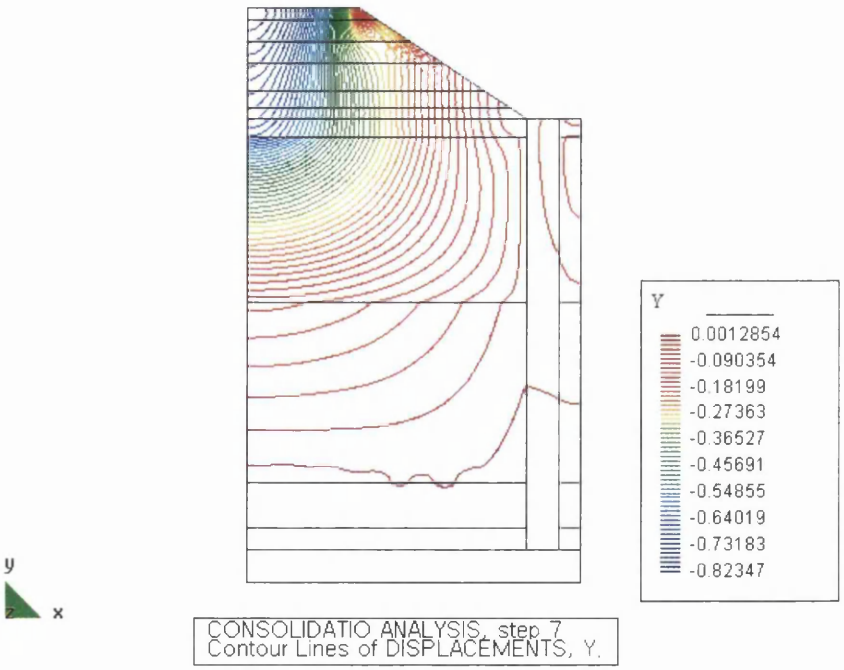
(a) y-displacement of time step 1 and no embankment.



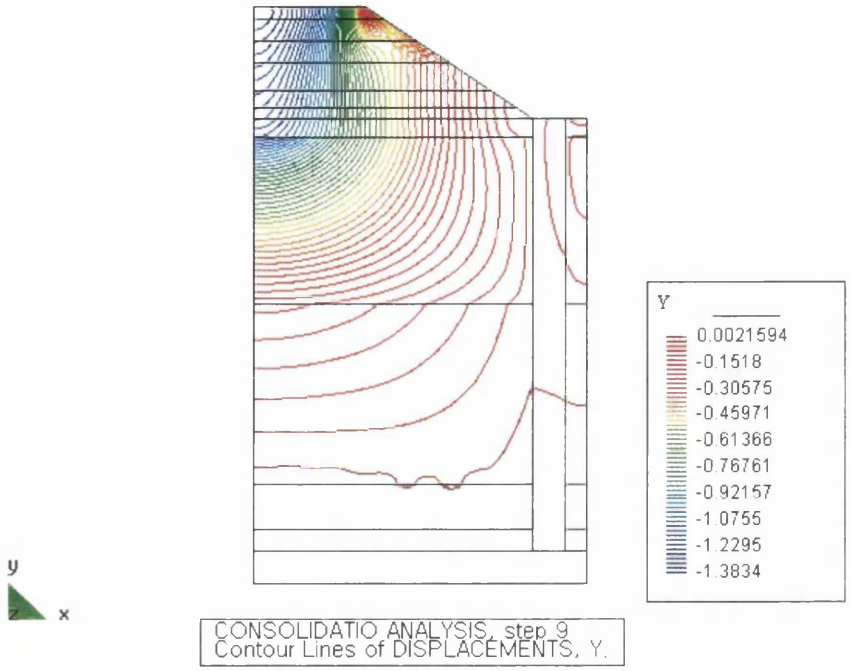
(b) y-displacement of time step 3 and embankment step 1.



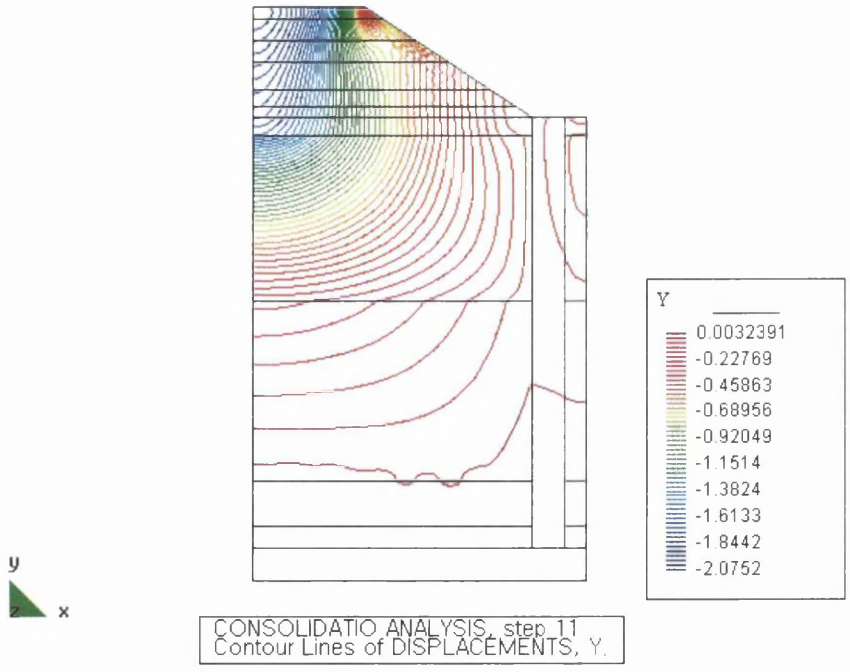
(c) y-displacement of time step 5 and embankment step 2.



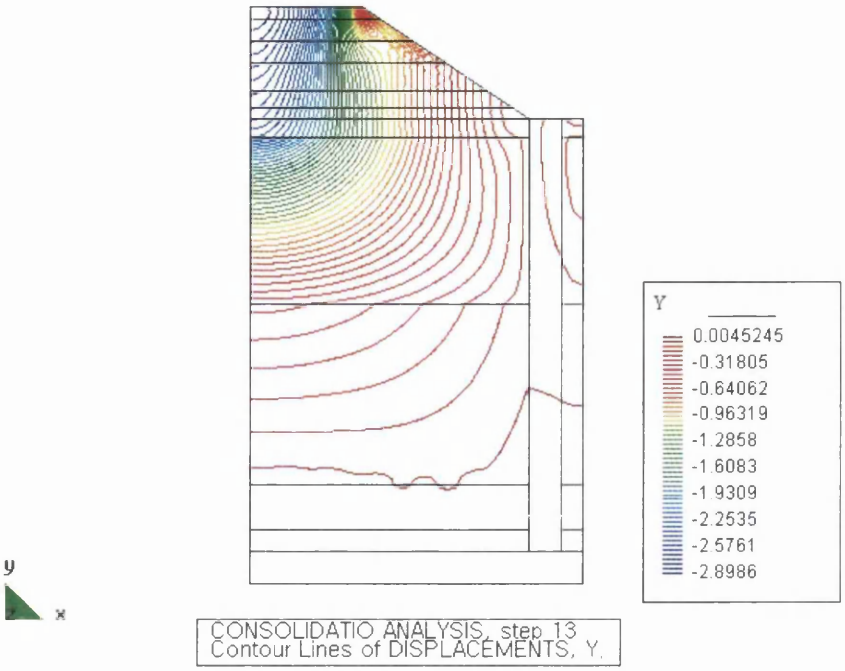
(d) y-displacement of time step 7 and embankment step 3.



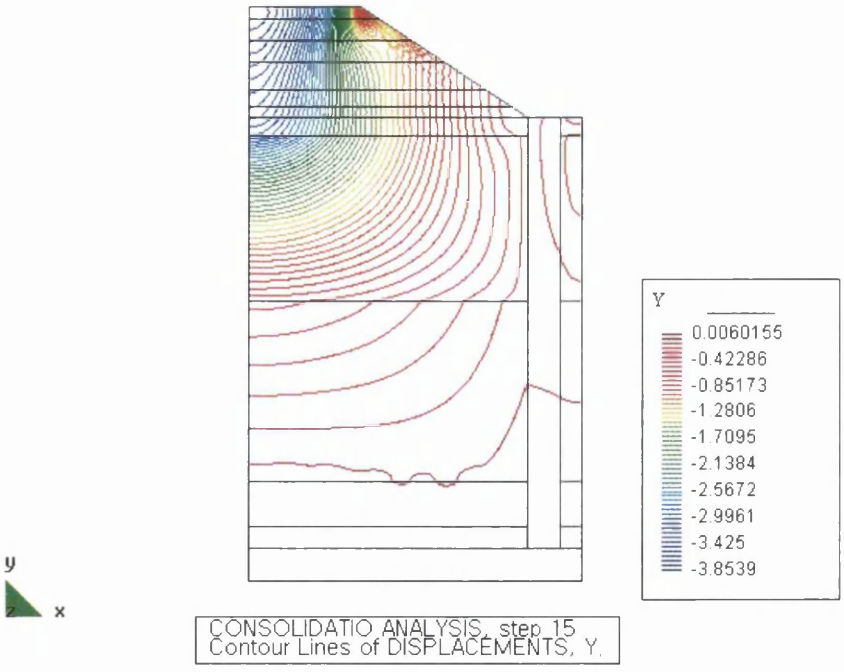
(e) y-displacement of time step 9 and embankment step 4.



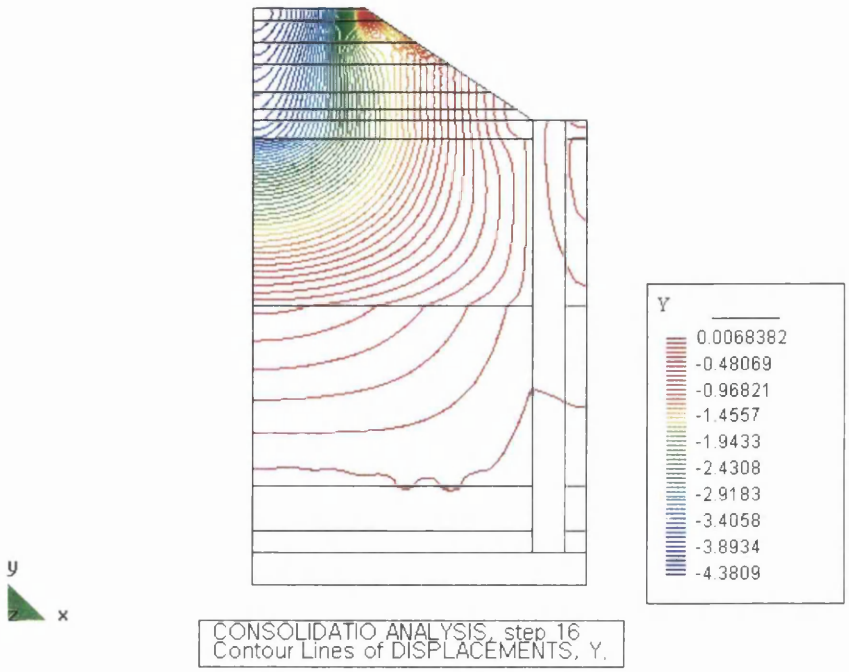
(f) y-displacement of time step 11 and embankment step 5.



(g) y-displacement of time step 13 and embankment step 6.



(h) y-displacement of time step 15 and embankment step 7.



(i) y-displacement of time step 16 and removing last embankment.

Figure 9.7 Contour plots for time and loading steps.

The results of contour plots of vertical settlements for the time step 1, 3, 5, 7, 9, 11, 13, 15 and 16. The loading of embankment steps are 1, 2, 3, 4, 5, 6, and 7. The last time and loading step is removing last embankment as shown as Figure 9.7.

9.5 Results of model analyses and field measurements.

The field measurements and results of modelling are shown as Figure 9.7. The comparison result is only y-direction settlements, which is useful to predict displacements and pore pressure. The numerical results are not exactly the same as field measurements because the material properties of sub-ground are not exact in field circumstance. To get more exact results, further experimental data are needed.

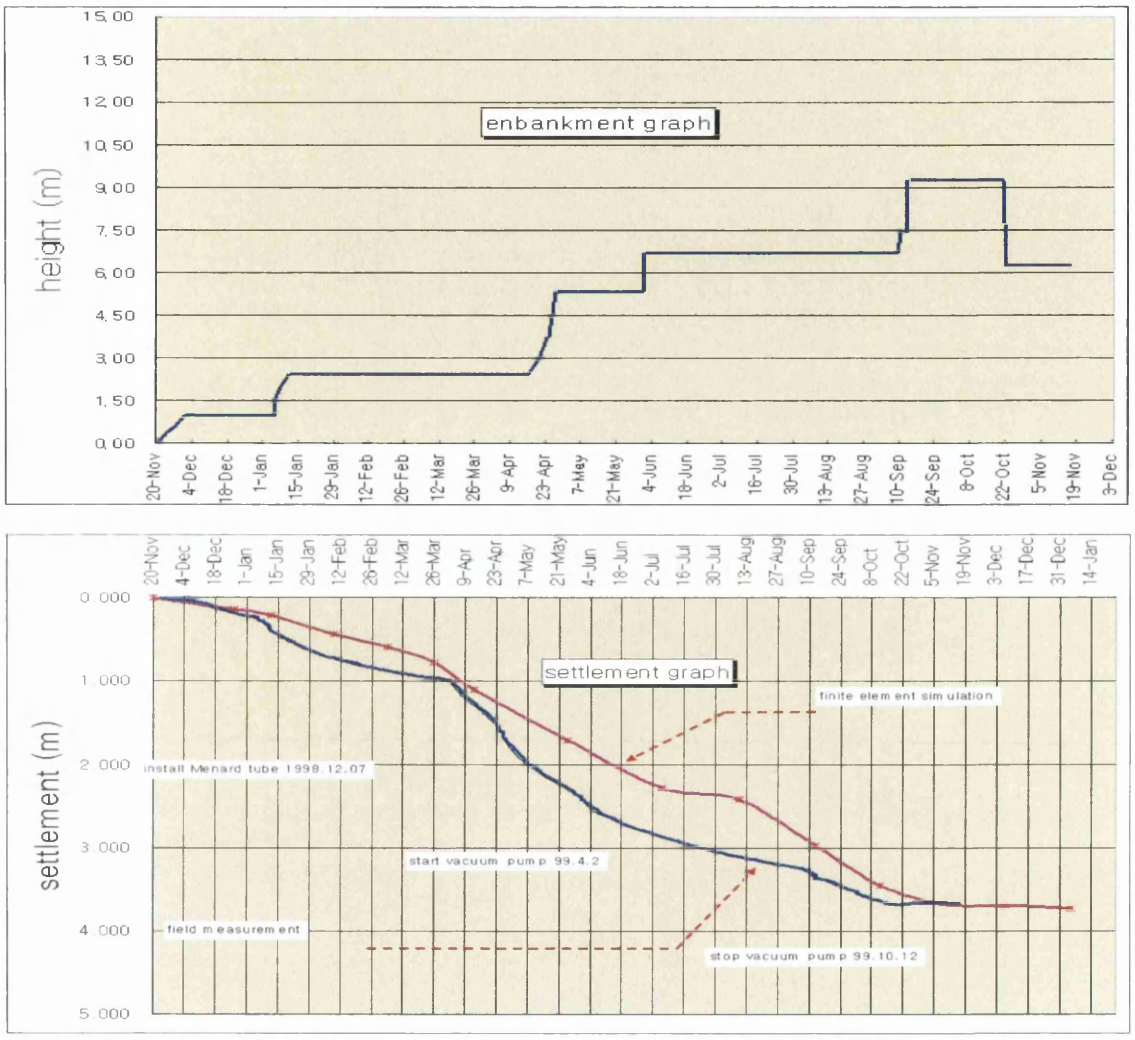


Figure 9.8. Comparison of modelled vertical displacements with filed measurements.

9.6 Concluding remarks

In this chapter, an application of the finite element modelling due to pumpage has been compared with practical field measurements at the Jangyoo sewage plant. The results of finite element analyses by pumpage are useful to the application of problems such as weak sub-soil improvements (Menard Vacuum System). The method developed in this thesis can be applied to other sub-soil improvement methods, such as sand drain, pack, and prefabricated vertical drain methods, with some modification.

Chapter 10

Conclusions and future research.

10.1 General discussion and conclusions

The complexity of physical soil properties and subsoil conditions means that every engineering project has its unique aspects. As has been shown in the literature review and discussion of founding theories of consolidation in earlier chapters, practical knowledge of the behavior of embankments, foundations and other structures involved in construction has been greatly advanced by the use of computers and the concomitant development of numerical methodology. Indeed, there has now been a shift from basic problem solving to the problem of modelling observed phenomena for predictive purposes.

Analysis of consolidation problem for soil improvement methods such as prefabricated, pack, and Menard drain method use Terzaghi's one-dimensional calculation and Barron's vertical drain method only. When using the finite element method, average of material properties is used to calculate settlement of ground and the time required.

This thesis is concerned with the development of a practical tool for consolidation analyses, specifically, the problem of continuum modelling of vacuum-induced consolidation due to pumpage. Classic mechanics alone does not provide sufficient information on the global motion equation of a fluid in porous media. The ideas and numerical simulation developed for this thesis have particular application to coupled isothermal porous continua.

Changing values of interior boundary conditions at pore-pressure terms is a critical element of the finite element modelling and numerical simulation presented in this thesis. The blocked profile solver with separated routine for fixed boundary condition is also contains step solution method for every embankment stage. As well it can be used to

change boundary condition for starting and stopping pumpage of pore water during construction.

The result of the model to apply field construction at Jang-yoo swage plant comparing with field measurements is useful for the practical purpose to predict the settlements and pore pressure. The results are not exact values of calculated finite continuum modelling because the properties of material are not exact in their natural states, so that more exact experiments needed to analyses.

10.2 Future research

There is now evidence that finite element modelling is well suited to the analyses of weak underground improvement methods, such as the Menard vacuum system. There is also reason to believe that it will prove useful for pack and pre-fabricated drain methods with some modifications.

The next step toward practical application is to develop three-dimensional program codes for the model and apply these to pack or prefabricated drain methods. As well, further research aimed at multiphase flow will be useful for petroleum engineering problems, such as subsidence due to pumping oil or natural gas.

It is hoped that the work presented here does not mark the end of research, but the beginning of an even more ambitious and satisfying exploration into the continuum modelling of consolidation due to pumpage.

References

[1] J.E. Akin, *Finite Elements for Analysis and Design*, Academic Press (1994).

[2] A. Asaoka, 'Observational Procedure of Settlement Prediction', Soils and Foundations, Vol.18, No 4, 87-101 (1978).

[3] P.K. Barnerjee, *The boundary element methods in engineering*, McGraw-Hill (1994).

[4] R.A. Barron, "Consolidation of fine grained soils by drain wells", Trans. ASCE, Vol. 113, 718-754 (1948).

[5] K.J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice Hall (1996).

[6] J. Bear, *Dynamics of Fluids in Porous Media*, American Elsevier, New York (1972).

[7] J. Bear and Y. Bachmat, 'Transport phenomena in porous media', ed. J. Bear and M. Y. Corapcioglu, Nato A. S. I. Series, E 82, Nijhoff, Dordrecht, 3-62 (1984).

[8] M.A Biot, 'General theory of three-dimensional consolidation', J. Appl. Phys., 12, 155-64 (1941).

[9] M.A. Biot, 'Theory of elasticity and consolidation for a porous anisotropic solid', J. Appl. Phys., 26, 182-5 (1955).

[10] M.A. Biot, 'Theory of deformation of a porous viscoelastic anisotropic solid', J. Appl. Phys., 27, 452-69 (1956).

[11] M.A. Biot, 'General theory of three-dimensional consolidation', J. Appl. Phys., 12, 155-164 (1941).

[12] C.B. Brown and I.P. King, 'Automatic embankment analysis: equilibrium and stability conditions', Geotechnique 16(3), 209-19 (1966).

[13] S.E. Buckley and M.C. Leverett, 'Mechanisms of fluid displacements in sands', Trans. AIME, 146, 108-116 (1942).

[14] N. Carillo, 'Simple two and three dimensional cases in the theory of consolidation of soils', Journal of Mathematics and Physics. Vol. 21, No 1.

[15] CIMNE, *GiD USER MANUAL*, CIMNE, (2002).

[16] V. Choa, 'Drains and vacuum preloading pilot test', Proc. 12th ICSMFE, Rio de Janeiro, 1347-1350 (1989).

[17] R.W. Clough, and Y.Rashid, 'Finite element analysis of axi-symmetric solids', J. Eng. Mech. Div., ASCE, 91(EM1), 71-85 (1965).

[18] J.M. Cognon, 'Vacuum consolidation', France Geotech Vol. 57, 37-47 (1991).

[19] J.M. Cognon, I.Juran and S.Thevanayagam. 'Vacuum Consolidation Technology - Principles and Field Experience', proceedings of 1994 conference on foundations and embankments deformations held June 16-18, 1994, College station of Texas (1994).

[20] J.M. Cognon, 'Menard Vacuum Consolidation'. Internal Document (1994).

[21] N.H. Cole and J.B. Burand, 'Observations of retaining wall movements associated with large excavations', Proc. 5th Europ. Conf. Soil Mech. Found. Eng., Madrid (1972).

[22] R.E. Collins, *Flow of Fluids through Porous Materials*, Reinhold, New York (1961).

[23] A.T. Corey, 'The interrelation between gas and oil relative permeabilities', Producers Monthly, 19, 38-41 (1954).

[24] B.C. Craft and M.F. Hawkins, *Applied Petroleum Reservoir Engineering*, Prentice-Hall, Englewood Cliffs (1969).

[25] H.B. Crichlow, *Modern Reservoir Engineering a Simulation Approach*, Prentice-Hall, Englewood Cliffs (1977).

[26] Dae Yoo Co. Design report of weak soil improvement for Jangyou sewage plant, (1998. 4), (Korean).

[27] P. De Saint Simon and Y. Rodriguez, "Surcharge preloading: the vacuum consolidation method versus wick drains." Southeastern transportation geotechnical engineering conference, Atlanta, Georgia. (1991)

[28] R.J.M. De Wiest, 'On the storage coefficient of groundwater flow', J. Geophys. Res., 71(4), 1117-22 (1966).

[29] R.J.M. De Wiest, *Flow through Porous Media*, Academic Press, New York (1969).

[30] W. James Demmel, R. John Gilbert and Xiaoye S. Li, ' SuperLU Users Guide', Computer Science Division, University of California, Berkeley (1999).

[31] C.S. Desai, *Numerical method in geotechnical engineering*, McGraw-Hill (1977).

[32] C.S. Desai, 'Non-linear analysis using spline functions', J. Soil Mech. Found. Eng. Div., ASCE, 97(SM10), 1461-80 (1971).

[33] Gouri Dhatt and Gilbert Touzot, *The finite element method displayed*, John Wiley and Sons, (1984).

[34] D.C. Drucker, 'A definition of stable inelastic material', Trans. A. S. M. E., 26, 101-6 (1959).

[35] A. Finol and S.M. Farouq Ali, 'Numerical simulation of oil production with simultaneous ground subsidence', S.P.E.J., 15, 411-24 (1975).

[36] G. Gambolati 'Equation for one-dimensional vertical flow of groundwater. 1. The rigorous theory', Water Resour. Research, 9, 1022-27 (1973).

[37] G. Gambolati, 'Equation for one-dimensional vertical flow of groundwater. 2. Validity range of the diffusion equation', Water Resour. Research, 9(5), 1385-95 (1973).

[38] G. Gambolati, 'Second-order theory of flow in three-dimensional deforming media', Water Resour. Research, 10, 1217-7 (1974).

[39] G. Gambolati and R.A. Freeze, 'Mathematical simulation of the subsidence of Venice. 1. Theory', Water Resour. Research, 9, 721-33 (1973).

[40] G. Gambolati, P. Gatto and R. A. Freeze, 'Mathematical simuation of the subsidence of Venice. 2. Results', Water Resour. Research, 10(3), 563-77 (1974).

[41] J. Geertsma, 'Problems of rock mechanics in petroleum production engineering', Proc. 1st Conf. Int. Soc. Rock Mech., Lisbon, 1, 585-94 (1966).

[42] D.V. Griffiths and I.M. Smith, *Numerical Methods for Engineers*, Blackwell Scientific Publications, (1991).

[43] S. Hansbo, 'Consolidation of clay by band shaped prefabricated drains', Ground Engineering, Vol. 3, No. 5. (1981).

[44] B. Hoyaux and B. Landanyi, 'Cavitational stress field around a tunnel in soft clay', Can. Geotech. J. (7), 54-61 (1970).

[45] C. Humpheson, Finite Element Analysis of Elasto-visco-Plastic Soils, Ph. D. Thesis, University College of Swansea (1976).

[46] C.T. Hwang, N.R. Morgenstern, and D.W. Murray, 'On solution of plane strain consolidation problems by finite element method', Canadian Geotech. J.(8), 109-18 (1971).

[47] C.W. Ihm and F. Masse, 'Successful application of Menard Vacuum consolidation method to Nakdong river soft clay in Kimhae, South Korea.', Menard Co. Ltd. (1996).

[48] J.S. Kang, 'A study of clayey ground deformation due to embankment', Theses collection of Kyungnam University 12, 185-197, (1994).

[49] F.H. Kulhawy and J. M. Duncan, 'Stresses and movements in Oroville dam', Proc. ASCE., 98(SM7), 653-65 (1972).

[50] Leon Lapidus and George F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley & Sons (1999).

[51] M.C. Leverett, 'Capillary behaviour in porous media', Petr. Trans., AIME, 142, 341-58, (1941).

[52] R.W. Lewis and B.A. Schrefler, *The Finite Element Method in the Deformation and Consolidation of Porous Media*, John Wiley & Sons (1987 & 1998).

[53] R.W. Lewis, and B.A. Schrefler, 'A fully coupled consolidation model of the subsidence of Venice', Water Res. Research, 14, 223-30, (1978).

[54] R.W. Lewis, and B.A. Schrefler, 'A finite element simulation of the subsidence of a gas reservoir undergoing a waterdrive', in Finite Elements in Fluids, Vol. 4, ed. R.H. Gallagher et al., Wiley, London, 179-200 (1982).

[55] R.W. Lewis, G.K. Roberts, and O.C. Zienkiewicz, 'A non linear flow and deformation analysis of consolidation problemns', Proc. 2nd Int. Conf. Num. Meth. Geom., Blacksburg. 1106-18, (1976).

[56] K.G. Manoj and S.K. Bhattacharyya, 'A Blocked Solver for Large, Unsymmetric, Sparce, Banded Matrix with Symmetric Profile', International Journal of Numerical Methods in Engineering, 40 3279-3295, (1997).

[57] J.L. Monte, and R. J. Kritzen, 'One dimensional mathematical model for large-strain consolidation', Geotechnique, 26(3), 495-510 (1976).

[58] G.C. Nayak, and O.C. Zienkiewicz, 'Convenient form of stress

invariants for plasticity', Proc. ASCE, 98(ST4), 949-53 (1972).

[59] D.J. Naylor, 'Non-linear Finite Element Models for Soils', Ph.D. Thesis, University College of Swansea (1975).

[60] D.J. Naylor, G.N. Pande, B. Simpson, and R. Tabb, 'Critical state model', Ch. 7, in Finite Elements in Geotechnical Engineering, Pineridge Press, Swansea (1981).

[61] S.P. Neuman, 'Finite element computer programs for flow in saturated-unsaturated porous media', Sec. Annual Rep., Proj, N. A10-SWC-77, Hydr. Eng. Lab., Haifa, Israel (1972).

[62] S.P. Neuman, 'Galerkin approach to saturated-unsaturated flow in porous media', in Finite Elements in Fluids, Vol. 1, ed. R. H. Gallagher, J. T. Oden, C. Taylor and O. C. Zienkiewicz, Wiley, London, 201-17, (1975).

[63] V.A. Norris, 'The Elasto-plastic Analysis of Soil Consolidation with Special Reference to Kinematic Hardening', Ph.D. Thesis, University College of Swansea (1980).

[64] A.D. M. Penman, J. B. Burland, and J. A. Charles, 'Observed and predicted deformations in a large embankment dam during construction', Proc. Inst. Civ. Engrs., 49, 1-21, (1971).

[65] B. Pezot, 'Intermediate report of vacuum consolidation - Kwangyang Container Terminal project' Internal document (1994).

[66] PLAXIS B. V., PLAXIS(Finite Element Code for Soil and Rock Analyses) A. A. Balkema (1998)

[67] F.B. Plummer et al. 'American Petroleum Institute Drilling Production and Practice', American Petroleum Institute, (1937).

[68] K.H. Roscoe, and J.B. Burland, 'On the generalized stress-strain behaviour of "wet" clay', Engineering Plasticity, Cambridge Univ. Press, pp. 535-609, (1968).

[69] K.H. Roscoe, and H.B. Poorooshasb, 'A theoretical and experimental study of strains in triaxial compression tests on normally consolidated clays', Geotechnique. 13(1), 12-34, (1963).

[70] K.H. Roscoe, A.N. Schofield, A. Thurairajah, 'Yielding of clays in states wetter than critical', Geotechnique, 13(3), 211-40, (1963).

[71] N.M. Safai, and G.F. Pinder, 'Vertical and horizontal land deformation in a desaturating porous medium', Adv. Water Res., 2, 19-25 (1979).

[72] R.S. Sandhu, 'Fluid Flow in Saturated Porous Elastic Media', Ph.D. Thesis, Univ. of Calif., Berkeley (1968).

[73] R.S. Sandhu and E.L. Wilson, 'Finite element analysis of seepage in elastic media', Proc. ASCE, J. Eng. Mech. Div., 95(EM3), 641-52, (1969).

[74] Sangjee Menard Co. *Consolidation report phase 3*, (1999. 12).

[75] Sangjee Menard Co. *Report of improvement for Jangyoo sewage plant*, (2000. 11), (Korean).

[76] Sangjee Menard Co. *Report of vacuum consolidation for Jangyoo sewage plant*, (2000. 4), (Korean).

[77] A. Scheidegger *The Physics of Flow Through Porous Media*, Univ, of Toronto Press (1957).

[78] A. N. Schofield and C.P. Wroth, *Critical State Soil Mechanics*, McGraw-Hill (1968).

[79] B. A. Schrefler, 'The Finite Element Method in Soil Consolidation (With Applications to Surface Subsidence)', Ph.D. Thesis, University College of Swansea (1984).

[80] B.A. Schrefler, R.W. Lewis and C. E. Majorana, 'The problem of rebound in subsidence models', in Finite Element Flow Analysis, ed. T. Kawai, University of Tokyo Press, 689-96 (1982).

[81] B.A. Schrefler, R. W. Lewis and C.E Majorana, 'Subsidence above volumetric and waterdrive gas reservoirs', Int. J. Num. Meth. Fluids, 1, 101-15 (1981).

[82] H.J. Siriwardance and C. S. Desai, 'Two numerical schemes for nonlinear consolidation', Int. J. Num. Meth. Eng., 17, 405-26 (1981).

[83] H.C. Slider, *Practical Petroleum Reservoir Engineering Methods*, Peter. Pub. Comp., Tulsa (1976).

[84] I.M. Smith and D.V. Griffiths, *Programming the Finite Element Method*, 3rd edn., John Wiley & Sons (1998).

[85] M.M. Stabrowski, 'A Block Equation Solver for Large Unsymmetric Linear Equation systems with Dense Coefficient Matrix', International Journal for Numerical Methods in Engineering, 24, 289-300 (1987).

[86] M.B. Standing, *Volumetric and Phase Behaviour of Oil Field Hydrocarbon Systems*, Reinhold, New York (1952).

[87] D.W. Taylor *Fundamental of soil mechanics*, John Wiley & Sons

(1948).

[88] K. Terzaghi, *Theoretical Soil Mechanics*, Wiley, New York (1943).

[89] S. Whitaker, 'Advances in the theory of fluid motion in porous media', Ind. Eng. Chem., 61, 14-28 (1969).

[90] I.R. White, 'The Finite Element Analysis of Multiphase Flow in Porous Media and its Application to Reservoir Engineering', Ph.D. Thesis, University College of Swansea (1978).

[91] O.C. Zienkiewicz, C. Humpheson and R. W. Lewis, 'A unified approach to soil mechanics', Ch. 4 in Finite Elements in Geomechanics, ed. G. Gudehus, Wiley, London (1977).

[92] O.C. Zienkiewicz, *The Finite Element Method*, McGraw-Hill, London (1977).

[93] O.C. Zienkiewicz, C. Humpheson and R. W. Lewis, 'Associated and nonassociated viscoplasticity in soil mechanics', Geotechnique, 25(4), 671-89 (1975).

[94] O.C. Zienkiewicz, S. Valliappan, and I.P. King, 'Elasto-Plastic Solutions of engineering problems: initial stress finite element approach', Int. J. Num. Meth. Eng., 1, 75-100, (1968).

[95] O.C. Zienkiewicz, *The Finite Element Method*, 4rd edn, McGraw-Hill (1991).

Appendix 1

Main program

```
PROGRAM RPLASCON
!*****************************************************************
!     2005-03-17 THUSDAY
!     CHECKED 01 MARCH, 2004 MONDAY. (TOTAL DISPLACEMENTS FOR TIME
!                                     STEPS)
!     20:30 AT 17 OCTOBER, 2003 FRIDAY. (19:00)
!     FOR FORTRAN90-95
!     MUST BE ADD FOR GRAVITY LOAD (SELF LOAD)
!     CALCULATE ELEMENT GRAVITY NODAL POINT LOAD AT
!      GVLOAD (NELEM,NEVAB) AND ADD CONSTRUCTION LAYER.
!      OMMIT GROUND LAYER.
!     ASSIGN UNIFORM LOAD NODAL POINT BY GID GRAPHIC AND CALCULATE
!      NODAL POINTS LOAD.
!*****************************************************************
!     05 AUGUST 2003(00:12) CONSTRUCTION LAYER CONTAINED
!                          MUST REVISE DIRICH AND UNSOL SUBROUTINE
!     04 AUGUST 2003(16:41) EXACT FOR PUMPING CONSOLIDATION
!     BOUNDARY CONDITIONS AT INTERIA PORE PRESSURE(EXACT)
!     REVISED AT 08 AUGUST, 2003 (21:25)
!     24 AUGUST, 2003 (16:32) MUST BE INCLUDED
!     GRAVITY WEIGHT
!     GRAPHICS FOR LOAD STEP, STRESS
!*****************************************************************
!     THIS PLASCON IS CORRECT OF FRONTAL AND BLOCKED PROFILE
!     SOLVER FOR CONSOLIDATION PROBLEM.
!     THIS NEED STAGGED BLOCKED PROFILE SOLVER FOR THERMO-
!          CONSOLIDATION PROBLEM
!     13 JANUARY, 2003
!     PROFESSOR JAE SUN   KANG.
!     CIVIL ENGINEERING DEPARTMENT, ENGINEERING COLLEGE.
!     KYUNG NAM UNIVERSITY
!     449 WOEL YOUNG DONG, MASAN CITY, KYUNG NAM.
!     SOUTH KOREA. (631-701)
!     E-MAIL :  NUSEAJ@KYUNGNAM.AC.KR
!     PHONE  :  82-55-249-2669
!*****************************************************************
!     CONVINCED AT UNIVERSITY OF WALES, SWANSEA.
!     DATA FILE ::
!          EXCON.DAT (KIM HAE TRIAL EMBANKMENT-CONSOLIDATION)
!          EXTH.DAT   (TEXT EXAMPLE OF THERMO-CONSOLIDATION)
!     CONSOLIDATION PROBLEM
!          (FRONTAL SOLVER AND PROFILE SOLVER EQUAL)
!     THERMO-CONSOLIDATION PROBLEM (FRONTAL SOLVER ONLY)
!                                  (STAGGED PROFILE SOLVER NEED)
!                22ND JANUARY, 2003
!     BLOCKED SYMMETRIC AND UNSYMMETRIC SOLVER IS RIGHT.
!                15TH MARCH,    2003 (11:20)
!*****************************************************************
!     ARRAYS FOR
!     NODLAY(LNP)    =  LAYER NUMBER OF EVERY NODE
!     NELLAY(LNE)    =  LAYER NUMBER OF EVERY ELEMENT
!     NEQLAY(LNRHS) =  LAYER NUMBER OF EVERY EQUATION
!*****************************************************************
!     GID GRAPHICS FOR POST PROCESSING
```

```
!     NODDUP(NP)     = DUPLICATION NUMBERS OF NODES
!     NODNUM(NP)     = TEMPERALLY ARRAY OF NODE NUMBER FOR
!                      DUPLICATION
!     STNOP(NP,4)    = STRESSES FOR ALL NODES
!     SNNOP(NP,4)    = STRAINS FOR ALL NODES
!     SSTRNP(4,8)    = STRESSES OF ELEMENT NODAL POINTS
!     SSTNNP(4,8)    = STRAINS OF ELEMENT NODAL POINTS
!     SSTRGP(4,4)    = STRESSES OF ELEMENT GAUSS POINTS
!     SSTNGP(4,4)    = STRAINS OF ELEMENT GAUSS POINT
!     TRANS(4,4)     = MATRIX FOR TRANSFER GAUSS POINT VALUESES
!                      TO NODAL POINT
!*****************************************************************
      USE KCHECK_COM
      USE INPOUT_COM
      USE ITAPE_COM
      USE NSPAC_COM
      USE NSPAC1_COM
      USE NSPAC2_COM
      USE NSPAC3_COM
      USE NSPAC4_COM
      USE NSPAC5_COM
      USE ICONT_COM
      USE ICONT1_COM
      USE IPRINT_COM
      USE MPLAST_COM
      USE NSSOL_COM
      USE NFORT_COM
      USE ISYSC_COM
      USE RSPAC2_COM
      USE VAR_COM
      USE RINTG_COM
      USE FPLAST_COM
      USE GAUSS_COM
      USE CTIMS_COM
      USE CTMAS_COM
      USE WALEV_COM
!*****************************************************************
      IMPLICIT DOUBLE PRECISION (A - H, O - Z)
      CHARACTER (LEN=72) :: TITLE, DATCHK
      CHARACTER (LEN=4) :: TCOM
      CHARACTER (LEN=20) ::    &
          INFIL, OUTFL, FDDIS, FDSTR, FDSTN, FDRES,      &
          FDINS, RESTR, RESTN, FNODLAY, FNELLAY, FNEQLAY
      CHARACTER (LEN=80) :: DATCK
!*****************************************************************
      INTEGER, DIMENSION (:) ::  &
       IONARY, NBC,     NELM,    IFFIX,  JFFIX,  KFFIX,  MHIGH,      &
       MAXAJ,  ISTAG,   NCOLBV, ICOPL,  NDUM,   NODEL,  NPP,        &
       NN,     IMAT,    NFIX,   NSTEPT, NELF,   NLI,    NLT
      INTEGER, DIMENSION (:) :: NODLAY, NELLAY, NEQLAY, KCONST
      ALLOCATABLE ::    &
       IONARY, NBC,     NELM,    IFFIX,  JFFIX,  KFFIX,  MHIGH,      &
       MAXAJ,  ISTAG,   NCOLBV, ICOPL,  NDUM,   NODEL,  NPP,        &
       NN,     IMAT,    NFIX,   NSTEPT, NELF,   NLI,    NLT
      ALLOCATABLE :: NODLAY, NELLAY, NEQLAY, KCONST
!-----------------------------------------------------------------
!     GRAPHICS OF GID (10 NOV. 2003)
!-----------------------------------------------------------------
      INTEGER, DIMENSION (:) :: NODDUP, NODNUM
      ALLOCATABLE :: NODDUP, NODNUM
!-----------------------------------------------------------------
      INTEGER, DIMENSION (:,:) ::  &
       NOP,    IFPRE,  JFPRE,  LEQNS,  ISMOOT,  LPOIN, NODBON,  &
```

```
       IFGID,  IFORG
      ALLOCATABLE ::  &
       NOP,    IFPRE,  JFPRE,  LEQNS,  ISMOOT,  LPOIN, NODBON,   &
       IFGID,  IFORG
!-----------------------------------------------------------------------
      DOUBLE PRECISION, DIMENSION (:) ::  &
       R1,     U,      BIT4Q,  R2,     ABDUM,  STM1,   STM2,      &
       QHLOAD, STLOAD, PLOAD,  XG,     QLOAD,  ELPORE, SLOAD,     &
       DARDUM, P,      STRESS, CG,     STRES1, W,      STRINC,    &
       SK,     SL,     ALOAD,  ABOUN,  FIXED,  FIXEB,  FIXEC,     &
       STMOV,  TINCR,  ELF,    ELHF,   ELTEMP, RLEVEL, DENSL,     &
       STDUM,  DELR,   DELT,   CORD2,  TFAC,   TFACEL, STARTV,    &
       REACT,  XCORA,  YCORA
      ALLOCATABLE ::  &
       R1,     U,      BIT4Q,  R2,     ABDUM,  STM1,   STM2,      &
       QHLOAD, STLOAD, PLOAD,  XG,     QLOAD,  ELPORE, SLOAD,     &
       DARDUM, P,      STRESS, CG,     STRES1, W,      STRINC,    &
       SK,     SL,     ALOAD,  ABOUN,  FIXED,  FIXEB,  FIXEC,     &
       STMOV,  TINCR,  ELF,    ELHF,   ELTEMP, RLEVEL, DENSL,     &
       STDUM,  DELR,   DELT,   CORD2,  TFAC.   TFACEL, STARTV,    &
       REACT,  XCORA,  YCORA
      DOUBLE PRECISION, DIMENSION (:) :: APREDI, BB
      ALLOCATABLE :: APREDI, BB
!-----------------------------------------------------------------------
      DOUBLE PRECISION, DIMENSION (:,:) ::  &
       PROP,   CORD,   SMAT,   COC,    FLH,    FLG,    COM,       &
       PLK,    TLH,    COTU,   TLL,    TLG,    TSL,    VEL,       &
       RILOAD, RLOAD,  B,      D,      DB,     BD,     CEL,       &
       DEL,    CW,     DW,     PDUM,   DIS,    STRAIN, ST,        &
       BVALUE, TTDIS,  RGRAV,  PFIXED, FIXVAL, PUFIX
      ALLOCATABLE ::  &
       PROP,   CORD,   SMAT,   COC,    FLH,    FLG,    COM,       &
       PLK,    TLH,    COTU,   TLL,    TLG,    TSL,    VEL,       &
       RILOAD, RLOAD,  B,      D,      DB,     BD,     CEL,       &
       DEL,    CW,     DW,     PDUM,   DIS,    STRAIN, ST,        &
       BVALUE, TTDIS,  RGRAV,  PFIXED, FIXVAL, PUFIX
!-----------------------------------------------------------------------
!     GRAPHICS OF GID  (10 NOV. 2003)
!-----------------------------------------------------------------------
      DOUBLE PRECISION, DIMENSION (:,:) ::  &
       STNOP,  SNNOP,  SSTRNP, SSTNNP, SSTRGP, SSTNGP, TRANS
      ALLOCATABLE ::  &
       STNOP,  SNNOP,  SSTRNP, SSTNNP, SSTRGP, SSTNGP, TRANS
!-----------------------------------------------------------------------
      DOUBLE PRECISION, DIMENSION (:,:,:) ::  &
       GASHT,  DBDUM,  BDUM,   DEN
      ALLOCATABLE :: GASHT,  DBDUM,  BDUM,   DEN
      DOUBLE PRECISION, DIMENSION (:,:,:) :: SWEIG
      ALLOCATABLE :: SWEIG
!-----------------------------------------------------------------------
!     ALLOCATABLE ARRAYS OF BLOCKED GLOBAL STIFFNESS
!-----------------------------------------------------------------------
      DOUBLE PRECISION, DIMENSION (:) ::  &
       STIFU,  STIFL,  STIFFU, STIFFL, DIAGN
      ALLOCATABLE :: STIFU,  STIFL,  STIFFU, STIFFL, DIAGN
!*********************************************************************
!     ARRAYS FOR
!     NODLAY(LNP)    =  LAYER NUMBER OF EVERY NODE
!     NELLAY(LNE)    =  LAYER NUMBER OF EVERY ELEMENT
!     NEQLAY(LNRHS) =  LAYER NUMBER OF EVERY EQUATION
!     SWEIG(NLAY,NE,NSIZE1) = LOAD OF CONSTRUCTION LAYER
!*********************************************************************
      DATA TCOM / 'ENDC' /
```

```
!*****************************************************************
!      FOR CALCULATE TOTAL RUN TIME
!*****************************************************************
       ST4D    = 0.0
       ST6D    = 0.0
!-----------------------------------------------------------------------
       ICHK1   = 1
       LPF     = 2
       ITP10   = 20
!*****************************************************************
!      ASSIGN ELEMENT STRESS AND STRAIN TO UNIT31 AND UNIT32
!*****************************************************************
       ITP31   = 51
       ITP32   = 52
!*****************************************************************
!      CARTESAN OR DISCOIDAL ?
!      IF IT USE CARTESIAN COORDINATE ICART=1
!      IF IT USE DISCOIDAL COORDINATE ICART=0
!*****************************************************************
!CCC   WRITE(*,2222)
!CCC   READ (*,2223)   INFIL
!CCC   WRITE(*,2224)
!CCC   READ (*,2223)   OUTFL
!CCC   WRITE(*,2225)
!CCC   READ (*,    *)  NSOL
!CCC   WRITE(*,2226)
!CCC   READ (*,    *)  ICART
!*****************************************************************
       WRITE ( *, 2227)
       READ ( *, * ) LCOM
!*****************************************************************
       NSOL    = 2
       ICART   = 1
       INFIL   = 'tdgid01mast.dat'
       IF (LCOM == 1) OUTFL = 'tdgidint.out'
       IF (LCOM == 2) OUTFL = 'tdgidlah.out'
       IF (LCOM == 3) OUTFL = 'tdgidpro.out'
       IF (LCOM == 4) OUTFL = 'tdgidgnu.out'
       IF (LCOM == 5) OUTFL = 'tdgidnas.out'
!*****************************************************************
 2222 FORMAT(2X,'INPUT FILE NAME = ')
 2223 FORMAT(A20)
 2224 FORMAT(2X,'OUTPUT FILE NAME = ')
 2225 FORMAT(2X,   &
        'SOLUTION METHOD IF NSOL=1 IT IS GLOBAL METHOD',/, &
        '                  IF NSOL=2 IT IS ACTIVE COLUMN METHOD',/, &
        2X,'***** DO NOT USE NSOL=2 IF THE PROBLEM IS ', &
        2X,'NON-ISOTHERMAL PROBLEM ******', &
        /,2X,' NSOL = ')
 2226 FORMAT(2X,   &
        '** CARTESIAN OR DISCOIDAL ? **',/, &
        2X,'** IF DISCOIDAL ICART IS ZERO **',/, &
        2X,'** IF CARTESIAN ICART IS ONE **',/, &
        2X,'** INPUT ICART VALUE **',I5)
 2227 FORMAT(2X,   &
        2X,'** WHAT IS THE KIND OF COMPILER ?',/, &
        2X,'LCOM=1 IS INTEL COMPILER',/, &
        2X,'LCOM=2 IS LAHEY COMPILER',/, &
        2X,'LCOM=3 IS PRO    COMPILER',/, &
        2X,'LCOM=4 IS G90    COMPILER',/, &
        2X,'LCOM=5 IS NASOFTCOMPILER')
!*****************************************************************
!      FOR CALCULATE TOTAL RUN TIME
```

```
!*********************************************************************
      ST4D      = 0.0
      ST6D      = 0.0
!---------------------------------------------------------------------
      CALL STCND (TMART, LCOM)
!---------------------------------------------------------------------
      IIN       = 25
      IOUT      = 26
!*********************************************************************
!     07 AUGUST, 2003 (01:00)
!     IODIS = DISPLACEMENTS FILE
!     IOSTR = STRESSES FILE
!     IOSTN = STRAINS FILE
!     IOGID = GID POST PROCESSING DATA FILE
!     IOINS = INITIAL STRESS FILE
!*********************************************************************
      IODIS     = 41
      IOSTR     = 42
      IOSTN     = 43
      IOGID     = 44
      IOINS     = 45
!*********************************************************************
!     09 AUGUST, 2003 (22:08)
!     IONODLAY    = LAYER NUMBER OF NODE FILE UNIT
!     IONELLAY    = LAYER NUMBER OF ELEMENT FILE UNIT
!     IONEQLAY    = LAYER NUMBER OF EQUATION FILE UNIT
!*********************************************************************
      IONODLAY = 46
      IONELLAY = 47
      IONEQLAY = 48
!*********************************************************************
!     07 AUGUST, 2003 (01:00)
!     FDDIS = DISPLACEMENTS FILE   NAME
!     FDSTR = STRESSES FILE   NAME
!     FDSTN = STRAINS FILE   NAME
!     FDRES = GID POST PROCESSING DATA FILE   NAME
!     FDINS = INITIAL STRESS FILE   NAME
!     RESTR = ELEMENT STRESS FILE FOR GID POST PROCESSING
!     RESTN = ELEMENT STRAIN FILE FOR GID POST PROCESSING
!*********************************************************************
      FDDIS     = 'rplas.dis'
      FDSTR     = 'rplas.str'
      FDSTN     = 'rplas.stn'
      FDRES     = 'rplas.flavia.res'
      FDINS     = 'rplas.ins'
      RESTR     = 'rplas-res.str'
      RESTN     = 'rplas-res.stn'
!*********************************************************************
!     09 AUGUST, 2003 (22:08)
!     FNODLAY = LAYER NUMBER OF NODE FILE NAME
!     FNELLAY = LAYER NUMBER OF ELEMENT FILE NAME
!     FNEQLAY = LAYER NUMBER OF EQUATION FILE NAME
!*********************************************************************
      FNODLAY   = 'nodlay.dat'
      FNELLAY   = 'nellay.dat'
      FNEQLAY   = 'neqlay.dat'
!---------------------------------------------------------------------
      ITP66     =  66
      ITP67     =  67
      ITP68     =  68
!*********************************************************************
!     UNIT FOR CHECKING FILE
!     8TH MAY 2003 (19:00)
```

```
!*********************************************************************
      KCHK1     = 31
      KCHK2     = 32
      KCHK3     = 33
      KCHK4     = 34
      KCHK5     = 35
      KCHK6     = 36
      KCHEK     = 37
!*********************************************************************
!     2005-03-22 TUESDAY
!     IGID  = 1 (FOR BOUNDARY CONDITIONS FROM GID DATA)
!     IBINT = 1 (FOR INTERIA BOUNDARY CONDITION OF PORE PRESSURE)
!     IBANK = 1 (FOR CONSTRUCTION ENBANKMENT)
!     04 AUGUST, 2003 (00:52)
!*********************************************************************
      IGID      = 1
      IBINT     = 1
      IBANK     = 1
!*********************************************************************
      ROOT3     = DSQRT (3.D0)
      NCOUNT    = 1
      NCN       = 8
      NSFR      = 2
   10 NW        = 1
      NR        = 2
!---------------------------------------------------------------------
      MTOTB     = 2000000
      MBLOCK    = 50
!---------------------------------------------------------------------
      MTN73     = 8000000
      MGFU      = 11000000
!---------------------------------------------------------------------
      ALLOCATE (BB(MTOTB))
!---------------------------------------------------------------------
      DO I = 1, MTOTB
        BB (I) = 0.D0
      END DO
!---------------------------------------------------------------------
      OPEN (IIN, FILE = INFIL, STATUS = 'OLD')
      OPEN (IOUT, FILE = OUTFL, STATUS = 'UNKNOWN')
!*********************************************************************
      OPEN (KCHK1, FILE = 'kchk1.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHK2, FILE = 'kchk2.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHK3, FILE = 'kchk3.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHK4, FILE = 'kchk4.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHK5, FILE = 'kchk5.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHK6, FILE = 'kchk6.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
      OPEN (KCHEK, FILE = 'kchek.rplas',   &
            FORM = 'FORMATTED', STATUS = 'UNKNOWN')
!*********************************************************************
!     07 AUGUST, 2003 (01:00)
!     OPEN FILES
!*********************************************************************
      OPEN (IODIS, FILE = FDDIS, STATUS = 'UNKNOWN')
      OPEN (IOSTR, FILE = FDSTR, STATUS = 'UNKNOWN')
      OPEN (IOSTN, FILE = FDSTN, STATUS = 'UNKNOWN')
      OPEN (IOGID, FILE = FDRES, STATUS = 'UNKNOWN',   &
```

```
            FORM = 'FORMATTED')
      OPEN (IOINS, FILE = FDINS, STATUS = 'UNKNOWN')
!*****************************************************************
      OPEN (ITP31, FILE = RESTR,   &
            FORM = 'UNFORMATTED', STATUS = 'UNKNOWN' )
      OPEN (ITP32, FILE = RESTN,   &
            FORM = 'UNFORMATTED', STATUS = 'UNKNOWN' )
!*****************************************************************
!     OPEN FILES FOR LAYER NUMBER OF NODES AND ELEMENTS
!*****************************************************************
      OPEN (ITP66, FILE = FNODLAY,   &
            FORM = 'FORMATTED', STATUS = 'OLD' )
      OPEN (ITP67, FILE = FNELLAY,   &
            FORM = 'FORMATTED', STATUS = 'OLD' )
      OPEN (ITP68, FILE = FNEQLAY,   &
            FORM = 'FORMATTED', STATUS = 'OLD' )
!*****************************************************************
!     09 AUGUST, 2003 (22:08)
!     FNODLAY = LAYER NUMBER OF NODE FILE NAME
!     FNELLAY = LAYER NUMBER OF ELEMENT FILE NAME
!     FNEQLAY = LAYER NUMBER OF EQUATION FILE NAME
!*****************************************************************
      WRITE (IOUT, 1001)
!-----------------------------------------------------------------
!     MASTER ICONTOL DATA
!-----------------------------------------------------------------
      READ (IIN, 8888) DATCK
      WRITE ( *, 8888) DATCK
   31 READ (IIN, 7) TITLE
      IF (TITLE == TCOM) GO TO 32
      WRITE (IOUT, 1002) TITLE
!-----------------------------------------------------------------
!     WRITE (KCHK1, 1002) TITLE
!-----------------------------------------------------------------
      GO TO 31
   32 READ (IIN, * )  &
       NTHERM, N, NSMITP, NSMITT, NINSCH, ISYMM
!-----------------------------------------------------------------
      WRITE (KCHK1, 7001)  &
       NTHERM, N, NSMITP, NSMITT, NINSCH, ISYMM
 7001 FORMAT( &
       1X,'NTHERM =',I5,1X,'N       =',I5,1X,'NSMITP=',I5,1X,/, &
       1X,'NSMITT =',I5,1X,'NINSCH=',I5,1X,'ISYMM =',I5)
!-----------------------------------------------------------------
      IF (NCOUNT == 1) NPROB = N
      IF (NTHERM > 0) GO TO 43
      NDF        = 3
      GO TO 44
   43 NDF        = 4
   44 READ (IIN, * )  &
       ITP1,   ITP2,   ITP3,   ITP4,   ITP5,   ITP6,   ITP7,       &
       ITP8,   ITP9,   ISTOP,  ISTORE, ICALC,  IVELOC, ICHECK
!-----------------------------------------------------------------
      WRITE (KCHK1, 7003)  &
       ITP1, ITP2, ITP3, ITP4, ITP5, ITP6, ITP7, &
       ITP8, ITP9, ISTOP, ISTORE, ICALC, IVELOC, ICHECK
 7003 FORMAT( &
       1X,'ITP1    =',I5,1X,'ITP2   =',I5,1X,'ITP3   =',I5,1X,/, &
       1X,'ITP4    =',I5,1X,'ITP5   =',I5,1X,'ITP6   =',I5,1X,/, &
       1X,'ITP7    =',I5,1X,'ITP8   =',I5,1X,'ITP9   =',I5,1X,/, &
       1X,'ISTOP   =',I5,1X,'ISTORE=',I5,1X,'ICALC =',I5,1X,/, &
       1X,'IVELOC =',I5,1X,'ICHECK=',I5)
!*****************************************************************
```

```
!      STAGGED PROFILE INDICATOR FOR THERMO-CONSOLIDATION
!*****************************************************************
       NSTAG     = 0
       IF (NSOL > 1.AND.NTHERM == 1) NSTAG     = 1
!*****************************************************************
       CALL FOPEN
!*****************************************************************
!      OPEN FILE NEEDED FOR PROFILE SOLUTION
!-----------------------------------------------------------------
       ITP11     = 21
       ITP12     = 22
       ITP13     = 23
!-----------------------------------------------------------------
       REWIND ITP1
       REWIND ITP2
       REWIND ITP3
       REWIND ITP4
       REWIND ITP5
       REWIND ITP6
       REWIND ITP7
       REWIND ITP8
       REWIND ITP9
       REWIND ITP10
!-----------------------------------------------------------------
!      MAIN ICONTOL DATA
!-----------------------------------------------------------------
       READ (IIN, * )  &
        NE,     NP,     NMAT,   NBN,    NSTEPS, NIL,    NLO,    &
        NELFL,  NITER,  NGAUS,  NTPLOT, NELSS,  NFPUMP, NLPUMP
!-----------------------------------------------------------------
       WRITE (KCHK1, 7005)    &
             NE, NP, NMAT, NBN, NSTEPS, NIL, NLO, NELFL, NITER, &
             NGAUS, NTPLOT, NELSS, NFPUMP, NLPUMP
 7005 FORMAT( &
        1X,'NE      =',I9,1X,'NP     =',I9,1X,'NMAT   =',I9,1X,/, &
        1X,'NBN     =',I9,1X,'NSTEPS=',I9,1X,'NIL    =',I9,1X,/, &
        1X,'NLO     =',I9,1X,'NELFL =',I9,1X,'NITER =',I9,1X,/, &
        1X,'NGAUS   =',I9,1X,'NTPLOT=',I9,1X,'NELSS =',I9,1X,/, &
        1X,'NFPUMP =',I9,1X,'NLPUMP=',I9)
!-----------------------------------------------------------------
       IF (NITER == 0) NITER = 2
       READ (IIN, * )  &
        IPSTR,  INDP,   ISTR,   IWRIT,  INTAP,  IOUTAP, IAUTOE, &
        IAUTON, NROW,   NCOL,   NLAYER
!-----------------------------------------------------------------
       WRITE (KCHK1, 7007)     &
             IPSTR, INDP, ISTR, IWRIT, INTAP, IOUTAP, &
             IAUTOE, IAUTON, NROW, NCOL, NLAYER
 7007 FORMAT( &
        1X,'IPSTR   =',I5,1X,'INDP   =',I5,1X,'ISTR   =',I5,1X,/, &
        1X,'IWRIT   =',I5,1X,'INTAP =',I5,1X,'IOUTAP=',I5,1X,/, &
        1X,'IAUTOE =',I5,1X,'IAUTON=',I5,1X,'NROW   =',I5,1X,/, &
        1X,'NCOL    =',I5,1X,'NLAYER=',I5)
!-----------------------------------------------------------------
       READ (IIN, * ) NNE, NNMAT, NNP
!-----------------------------------------------------------------
       WRITE (KCHK1, 7009)    &
             NNE, NNMAT, NNP
 7009 FORMAT( &
        1X,'NNE     =',I5,1X,'NNMAT =',I5,1X,'NNP    =',I5)
!-----------------------------------------------------------------
!      TO USE ARRAY SIZES
!-----------------------------------------------------------------
```

```
      NGNG      = NGAUS * NGAUS
      JIG       = 5 * NGNG
      NCOL      = NCOL + 1
      NPSR      = 0
      NPST      = 0
      IELEM     = 0
      INODE     = 0
      IELMAT    = 0
      IBOUND    = 0
      IDISP     = 0
      ISTRES    = 0
      INIT      = 0
!-----------------------------------------------------------------------
!     MAIN ICONTOL DATA
!-----------------------------------------------------------------------
      IF (INTAP /= 1) GO TO 46
      READ (IIN, * )  &
       IELEM,  IELMAT, INODE,  IBOUND, IDISP,  ISTRES, INIT
!-----------------------------------------------------------------------
      WRITE (KCHK1, 7011)   &
            IELEM, IELMAT, INODE, IBOUND, IDISP, ISTRES, &
            INIT
 7011 FORMAT( &
       1X,'IELEM  =',I5,1X,'IELMAT=',I5,1X,'INODE =',I5,1X,/, &
       1X,'IBOUND =',I5,1X,'IDISP =',I5,1X,'ISTRES=',I5,1X,/, &
       1X,'INIT   =',I5)
!-----------------------------------------------------------------------
   46 IF (NNE < 0) READ (IIN, * ) NRAD, NPSR, NTET, NPST, R1W
!-----------------------------------------------------------------------
      IF (NNE > 0)  NRAD = 0
      IF (NNE > 0)  NPSR = 0
      IF (NNE > 0)  NTET = 0
      IF (NNE > 0)  NPST = 0
      IF (NNE > 0)  R1W  = 0.0
!-----------------------------------------------------------------------
        IF (NNE.LT.0)  &
      WRITE (KCHK1, 7013) NRAD, NPSR, NTET, NPST, R1W
 7013 FORMAT( &
       1X,'NRAD   =',I5,1X,'NPSR  =',I5,1X,'NTET  =',I5,1X,/, &
       1X,'NPST   =',I5,1X,'R1W   =',E15.7)
!-----------------------------------------------------------------------
      NLEVEL = 0
      IF (ISTR == 2) READ (IIN, 7) DATCHK
      IF (ISTR == 2) READ (IIN, * ) NLEVEL, PLATK
!-----------------------------------------------------------------------
        IF (ISTR.EQ.2) WRITE (KCHK1, 7015) NLEVEL, PLATK
 7015 FORMAT( &
       1X,'NLEVEL =',I5,1X,'PLATK =',E10.3)
!-----------------------------------------------------------------------
!     TO USE ARRAY SIZES
!-----------------------------------------------------------------------
      IN        = JIG * NE
      KSIZE     = NCN * NDF
      KSIZE1    = NCN * 2
      KSIZE2    = NCN
      NRHS      = NP * NDF
      MRHS      = NRHS
      NSIZE     = KSIZE
      NSIZE1    = KSIZE1
      NSIZE2    = KSIZE2
      IF (NTHERM == 0)  LSIZE1      = 1
      IF (NTHERM == 1)  LSIZE1      = KSIZE1
      IF (NTHERM == 0)  LSIZE2      = 2
```

```
      IF (NTHERM == 1)  LSIZE2      = KSIZE2
!-----------------------------------------------------------------------
      MMAT      = NMAT
      MSTEPS    = NSTEPS
      MP        = NP
      ME        = NE
      MDF       = NDF
      IIG       = JIG
      MGNG      = NGNG
      JN        = IN
      MGAUS     = NGAUS
      MBN       = NBN
      ISTR1     = ISTR + ISTRES
      ICS1      = 4
      IF (IPSTR == 1) ICS1      = 3
      ICS2      = 2
      ICS1A     = ICS1
      ICS1B     = ICS1
!-----------------------------------------------------------------------
      MTNSL     = NSIZE1 * NSIZE2
!-----------------------------------------------------------------------
!     START ADRESS OF COMMON A-ARRAY
!-----------------------------------------------------------------------
!                                     PROP(NMAT,I18)
      I18   = 17
      IF (NTHERM > 0) I18   = 23
      ALLOCATE (PROP(NMAT,I18))
!                                     TINCR(NSTEPS)
!                                     NSTEPT(NSTEPP)
!                                     CORD(NP,2)
!                                     NOP(NE,8)
!                                     NODBON(NP,NDF)
!                                     LPOIN(NP,NDF)
!                                     IMAT(NE)
      NSTEPP = 1
      IF (NTPLOT /= 0) NSTEPP = NTPLOT
      ALLOCATE (TINCR(NSTEPS))
      ALLOCATE (NSTEPT(NSTEPP))
      ALLOCATE (CORD(NP,2))
      ALLOCATE (NOP(NE,NCN))
      ALLOCATE (NODBON(NP,NDF))
      ALLOCATE (LPOIN(NP,NDF))
      ALLOCATE (IMAT(NE))
!                                     NELM(NE1)
      NE1 = NE
      IF (NNMAT == 0) NE1 = 1
      ALLOCATE (NELM(NE1))
!                                     DELR(NRAD1)
      IF (NRAD > 0)   NRAD1 = NRAD
      IF (NRAD == 0)   NRAD1 = 1
      IF (NNP >= 0.OR.NPSR <= 0) NRAD1 = 1
      ALLOCATE (DELR(NRAD1))
!                                     DELT(NTET1)
      IF (NTET > 0)   NTET1 = NTET
      IF (NTET == 0)   NTET1 = 1
      IF (NNP >= 0.OR.NPST <= 0) NTET1 = 1
      ALLOCATE (DELT(NTET1))
!                                     CORD2(NROW)
!                                     GASHT(IGS,MDF,NP)
      IGS = 3
      MDF1 = 3
      IF (NINSCH == 3.OR.NINSCH == 2) IGS = 3
      IF (NINSCH == 3.OR.NTHERM == 1) MDF1 = 4
```

```
      IF (NINSCH == 2.OR.NTHERM == 1) MDF1 = 4
      ALLOCATE (CORD2(NROW))
      ALLOCATE (GASHT(IGS,MDF1,NP))
!                                     RLEVEL(NLEV1)
!                                     DENSL(NLEV1)
      NLEV1 = NLEVEL
      IF (ISTR /= 2) NLEV1 = 1
      ALLOCATE (RLEVEL(NLEV1))
      ALLOCATE (DENSL(NLEV1))
!                                     NLI(NIL1)
!                                     RILOAD(NIL1,NDF11)
      NIL1 = NIL
      IF (NIL == 0) NIL1 = 1
      NDF11 = NDF - 1
      IF (NTHERM == 1) NDF11 = NDF11 - 1
      IF (NIL == 0) NDF11 = 1
      ALLOCATE (NLI(NIL1))
      ALLOCATE (RILOAD(NIL1,NDF11))
!-----------------------------------------------------------------------
!     2005-03-22 TUESDAY FOR LOAD OF ENBANKMENT
!     IBANK = 0 DO NOT CONSIDER ENBANKMENT LOAD
!     IBANK = 1         CONSIDER ENBANKMENT LOAD
!-----------------------------------------------------------------------
!                                     NLT(NLO1)
!                                     RLOAD(NLO1,NDFA)
!                                     RGRAV(NP,2)
!                                     PFIXED(NP,NDF)
!-----------------------------------------------------------------------
!     1. NO ENBANKMENT
!-----------------------------------------------------------------------
      IF (IBANK == 0)   NLO1 = NLO
      IF (IBANK == 0 .AND. NTHERM == 1)   NDFA = NDF - 2
      IF (IBANK == 0 .AND. NTHERM == 0)   NDFA = NDF - 1
      IF (IBANK == 0 .AND. NLO == 0)   NLO1 = 1
      IF (IBANK == 0 .AND. NLO == 0)   NDFA = 1
!-----------------------------------------------------------------------
!     2. ENBANKMENT
!-----------------------------------------------------------------------
      IF (IBANK == 1)   NLO1 = NP
      IF (IBANK == 1 .AND. NTHERM == 1)   NDFA = NDF - 2
      IF (IBANK == 1 .AND. NTHERM == 0)   NDFA = NDF - 1
!-----------------------------------------------------------------------
      ALLOCATE (NLT(NLO1))
      ALLOCATE (RLOAD(NLO1,NDFA))
      ALLOCATE (RGRAV(NLO1,2))
      ALLOCATE (PFIXED(NP,NDF))
!-----------------------------------------------------------------------
!     2005-03-22 TUESDAY FOR LOAD OF ENBANKMENT
!-----------------------------------------------------------------------
!                                     TFAC(NSTEP1)
!-----------------------------------------------------------------------
      IF (IBANK == 0 .AND. NLO1 == 0)   NSTEP1 = 1
      IF (IBANK == 1 .AND. NLO1 > 0)    NSTEP1 = NSTEPS
      ALLOCATE (TFAC(NSTEP1))
!-----------------------------------------------------------------------
!                                     NELF(NELSS1)
!                                     ELF(NELSS1)
!                                     ELHF(NELSS2)
!                                     TFACEL(NSTEP2)
      NELSS1 = NELSS
      NSTEP2 = NSTEPS
      IF (NELSS == 0) NELSS1 = 1
      NELFL1 = NELSS1
```

```
!-----------------------------------------------------------------------
      IF (NELFL == 0) NSTEP2 = 1
      NELSS2 = NELSS1
      IF (NELSS == 0) NELSS2 = 1
      IF (NTHERM == 0) NELSS2 = 1
      NELFL2 = NELSS2
!-----------------------------------------------------------------------
      ALLOCATE (NELF(NELSS1))
      ALLOCATE (ELF(NELSS1))
      ALLOCATE (ELHF(NELSS2))
      ALLOCATE (TFACEL(NSTEP2))
!                                       BVALUE(NBN,NDF)
!                                       FIXVAL(NP,NDF)
!                                       PUFIX (NP,NDF)
!!!!                                    PNFIX (NP,NDF)
!                                       NFIX(NP)
!                                       IFGID(NBN,NDF)
!                                       NBC(NP)
      ALLOCATE (BVALUE(NBN,NDF))
      ALLOCATE (FIXVAL(NP,NDF))
      ALLOCATE (PUFIX (NP,NDF))
!!!!  ALLOCATE (PNFIX (NP,NDF))
      ALLOCATE (NFIX(NP))
      ALLOCATE (IFGID(NBN,NDF))
      ALLOCATE (NBC(NP))
!                                       ISMOOT(NSIZES,NSIZES)
      KSIZES = KSIZE2
      IF (NSMITP > 0.OR.NSMITT > 0) KSIZES = KSIZE2
      ALLOCATE (ISMOOT(KSIZES,KSIZES))
!                                       IONARY(NP)
      ALLOCATE (IONARY(NP))
!                                       STM1(JIG)
!                                       STM2(JIG)
      ALLOCATE (STM1(JIG))
      ALLOCATE (STM2(JIG))
!                                       R1(NRHS)
!                                       BIT4Q(NP)
      ALLOCATE (R1(NRHS))
      ALLOCATE (BIT4Q(NP))
!                                       SMAT(NSIZE,NSIZE)
!                                       COC(NSIZE1,NSIZE2)
!                                       FLH(NSIZE2,NSIZE2)
!                                       FLG(NSIZE2,NSIZE2)
!                                       COM(NSIZE1,NSIZE2)
!                                       PLK(NSIZE1,NSIZE1)
      ALLOCATE (SMAT(NSIZE,NSIZE))
      ALLOCATE (COC(NSIZE1,NSIZE2))
      ALLOCATE (FLH(NSIZE2,NSIZE2))
      ALLOCATE (FLG(NSIZE2,NSIZE2))
      ALLOCATE (COM(NSIZE1,NSIZE2))
      ALLOCATE (PLK(NSIZE1,NSIZE1))
!                                       COTU(LSIZE1,LSIZE2)
!                                       TLH(LSIZE2,LSIZE2)
!                                       TLL(LSIZE2,LSIZE2)
!                                       TLG(LSIZE2,LSIZE2)
!                                       TSL(LSIZE2,LSIZE2)
      ALLOCATE (COTU(LSIZE1,LSIZE2))
      ALLOCATE (TLH(LSIZE2,LSIZE2))
      ALLOCATE (TLL(LSIZE2,LSIZE2))
      ALLOCATE (TLG(LSIZE2,LSIZE2))
      ALLOCATE (TSL(LSIZE2,LSIZE2))
!                                       ELTEMP(LSIZE2)
!                                       QHLOAD(LSIZE2)
```

```
      ALLOCATE (ELTEMP(NSIZE2))
      ALLOCATE (QHLOAD(LSIZE2))
!                                     NODEL(NSIZE2)
      ALLOCATE (NODEL(NSIZE2))
!                                     QLOAD(NSIZE2)
!                                     ELPORE(NSIZE2)
!                                     SLOAD(NSIZE1)
!                                     STLOAD(NSIZE1)
!                                     PLOAD(NSIZE1)
      ALLOCATE (QLOAD(NSIZE2))
      ALLOCATE (ELPORE(NSIZE2))
      ALLOCATE (SLOAD(NSIZE1))
      ALLOCATE (STLOAD(NSIZE1))
      ALLOCATE (PLOAD(NSIZE1))
!                                     B(ICS1A,NSIZE1)
!                                     D(ICS1A,ICS1A)
!                                     BD(NSIZE1,ICS1A)
!                                     DB(ICS1A,NSIZE1)
      ALLOCATE (B(ICS1A,NSIZE1))
      ALLOCATE (D(ICS1A,ICS1A))
      ALLOCATE (BD(NSIZE1,ICS1A))
      ALLOCATE (DB(ICS1A,NSIZE1))
!                                     DBDUM(ICS1A,NSIZE1,NGNG)
!                                     BDUM(ICS1A,NSIZE1,NGNG)
!                                     DARDUM(NGNG)
!                                     PDUM(NSIZE2,NGNG)
      ALLOCATE (DBDUM(ICS1A,NSIZE1,NGNG))
      ALLOCATE (BDUM(ICS1A,NSIZE1,NGNG))
      ALLOCATE (DARDUM(NGNG))
      ALLOCATE (PDUM(NSIZE,NGNG))
!                                     P(NSIZE2)
!                                     W(NSIZE2)
      ALLOCATE (P(NSIZE2))
      ALLOCATE (W(NSIZE2))
!                                     CEL(2,NSIZE2)
!                                     DEL(2,NSIZE2)
      ALLOCATE (CEL(2,NSIZE2))
      ALLOCATE (DEL(2,NSIZE2))
!                                     CW(2,NSIZE2)
!                                     DW(2,NSIZE2)
      ALLOCATE (CW(2,NSIZE2))
      ALLOCATE (DW(2,NSIZE2))
!                                     STRAIN(NGNG,ICS1B)
!                                     ST(NGNG,6)
!                                     STRESS(ICS1B)
!                                     STRES1(ICS1B)
!                                     STDUM(ICS1B)
!                                     STRINC(ICS1B)
!                                     STARTV(ICS1B)
      ALLOCATE (STRAIN(NGNG,ICS1B))
      ALLOCATE (ST(NGNG,6))
      ALLOCATE (STRESS(ICS1B))
      ALLOCATE (STRES1(ICS1B))
      ALLOCATE (STDUM(ICS1B))
      ALLOCATE (STRINC(ICS1B))
      ALLOCATE (STARTV(ICS1B))
!                                     XG(NGAUS)
!                                     CG(NGAUS)
      ALLOCATE (XG(NGAUS))
      ALLOCATE (CG(NGAUS))
!                                     U(NB1=NBN*NDF)
      NB1 = NBN * NDF
      ALLOCATE (U(NB1))
```

```
!                                   DEN(2,NGNG,NE)
!                                   VEL(2,NSIZE2)
!                                   DIS(NDF,NP)
      NEA1 = NE
      IF (NTHERM == 0) NEA1 = 1
      ALLOCATE (DEN(2,NGNG,NEA1))
      ALLOCATE (NN(NSIZE2))
      ALLOCATE (VEL(2,NSIZE2))
      ALLOCATE (DIS(NDF,NP))
!---------------------------------------------------------------------
!     TTDIS (NP, NDF) IS TOTAL DISPLACEMENTS AT EACH TIME STEP
!     GID GRAPHICS RESULTS (22 FEB. 2004)
!---------------------------------------------------------------------
!                                   DIS(NDF,NP)
      ALLOCATE (TTDIS(NP,NDF))
!---------------------------------------------------------------------
!                                   SK(MTN73)
!                                   SL(MTNSL)
!                                   NPP(NP)
!                                   APREDI(NP)
!---------------------------------------------------------------------
      ALLOCATE (SK(MTN73))
      ALLOCATE (SL(MTNSL))
      ALLOCATE (NPP(NP))
      ALLOCATE (APREDI(NP))
!*********************************************************************
!     2005-03-26 SATURDAY
!     CONSTRUCTION LAYER ANALYSIS
!*********************************************************************
!                                   NODLAY(NP)
!                                   NELLAY(NE)
!                                   NEQLAY(NRHS)
!                                   KCONST(NSTEPS)
!                                   SWEIG(NLAY,NE,NSIZE1)
!---------------------------------------------------------------------
      ALLOCATE (NODLAY(NP))
      ALLOCATE (NELLAY(NE))
      ALLOCATE (NEQLAY(NRHS))
      ALLOCATE (KCONST(NSTEPS))
!*********************************************************************
!     POST PROCESSING GRAPHIC FOR GID (10, NOV 2003)
!*********************************************************************
!                                   NODDUP(NP)
!                                   NODNUM(NP)
!---------------------------------------------------------------------
      ALLOCATE (NODDUP(NP))
      ALLOCATE (NODNUM(NP))
!*********************************************************************
      ALLOCATE (NDUM(4))
!---------------------------------------------------------------------
!                                   STMOV(IN=5*NGAUS*NGAUS*NE)
!---------------------------------------------------------------------
      ALLOCATE (ALOAD(4))
      ALLOCATE (ABOUN(4))
      ALLOCATE (STMOV(IN))
!*********************************************************************
      ALLOCATE (ABDUM(4))
      IF (NSOL == 1) GO TO 8001
!---------------------------------------------------------------------
!     SOLVE FOR BLOCKED PROFILE SOLUTION
!---------------------------------------------------------------------
!                                   R2(MRHS=NP*NDF)
!                                   IFPRE(NP,NDF)
```

```
!                                       IFORG(NP,NDF)
!                                       JFPRE(NP,NDF)
!                                       FIXED(MRHS=NP*NDF)
!                                       FIXEB(MRHS=NP*NDF)
!                                       FIXEC(MRHS=NP*NDF)
!                                       LEQNS(NE,NSIZE)
!                                       IFFIX(MRHS=NP*NDF)
!                                       JFFIX(MRHS=NP*NDF)
!                                       KFFIX(MRHS=NP*NDF)
!                                       MHIGH(MRHS=NP*NDF)
!                                       MAXAJ(MRHS+ 1)
!                                       DIAGN(MRHS=NP*NDF)
!                                       ICOPL(MBLOCK)
!                                       NCOLBV(MBLOCK)
!                                       REACT(MRHS=NP*NDF)
!                                       XCORA(NP)
!                                       YCORA(NP)
!                                       STMOV(IN=5*NGAUS*NGAUS*NE)
!*****************************************************************
!     STAGGED SOLUTION INDICATOR VECTOR
!*****************************************************************
!                                       ISTAG(MRHS)
!*****************************************************************
 8001 CONTINUE
      IF (NSOL == 1) NPOIN = 1
      IF (NSOL == 1) NELEM = 1
      IF (NSOL == 1) NDOFN = 1
      IF (NSOL == 1) NNODE = 1
      IF (NSOL == 1) NEQNS = 1
      IF (NSOL == 1) NEQN1 = 1
      IF (NSOL == 1) NEVAB = 1
!-----------------------------------------------------------------
      IF (NSOL > 1)  NPOIN = NP
      IF (NSOL > 1)  NELEM = NE
      IF (NSOL > 1)  NDOFN = NDF
      IF (NSOL > 1)  NNODE = NCN
      IF (NSOL > 1)  NEQNS = NRHS
      IF (NSOL > 1)  NEQN1 = NRHS + 1
      IF (NSOL > 1)  NEVAB = KSIZE
!*****************************************************************
      ALLOCATE (R2(NEQNS))
      ALLOCATE (IFPRE(NPOIN,NDOFN))
      ALLOCATE (IFORG(NPOIN,NDOFN))
      ALLOCATE (JFPRE(NPOIN,NDOFN))
      ALLOCATE (FIXED(NEQNS))
      ALLOCATE (FIXEB(NEQNS))
      ALLOCATE (FIXEC(NEQNS))
      ALLOCATE (LEQNS(NELEM,NEVAB))
      ALLOCATE (IFFIX(NEQNS))
      ALLOCATE (JFFIX(NEQNS))
      ALLOCATE (KFFIX(NEQNS))
      ALLOCATE (MHIGH(NEQNS))
      ALLOCATE (MAXAJ(NEQN1))
      ALLOCATE (ICOPL(MBLOCK))
      ALLOCATE (NCOLBV(MBLOCK))
      ALLOCATE (REACT(NEQNS))
      ALLOCATE (XCORA(NP))
      ALLOCATE (YCORA(NP))
      ALLOCATE (ISTAG(NEQNS))
!*****************************************************************
!     POST PROCESSING GRAPHIC FOR GID (10, NOV 2003)
!*****************************************************************
      ALLOCATE (STNOP(NP,8))
```

```
      ALLOCATE (SNNOP(NP,8))
      ALLOCATE (SSTRNP(4,8))
      ALLOCATE (SSTNNP(4,8))
      ALLOCATE (SSTRGP(4,8))
      ALLOCATE (SSTNGP(4,8))
      ALLOCATE (TRANS(4,4))
!*****************************************************************
!     CLEAR ALL ARRAYS TO ZERO
!*****************************************************************
!     1. ONE DIMENSIONAL INTEGER ARRAYS
!-----------------------------------------------------------------
      CALL IZERO1 (IONARY, NP)
      CALL IZERO1 (NBC, NP)
      CALL IZERO1 (NELM, NE1)
      CALL IZERO1 (IFFIX, NEQNS)
      CALL IZERO1 (JFFIX, NEQNS)
      CALL IZERO1 (KFFIX, NEQNS)
      CALL IZERO1 (MHIGH, NEQNS)
      CALL IZERO1 (MAXAJ, NEQN1)
      CALL IZERO1 (ISTAG, NEQNS)
      CALL IZERO1 (NCOLBV, MBLOCK)
      CALL IZERO1 (ICOPL, MBLOCK)
      CALL IZERO1 (NDUM, 4)
      CALL IZERO1 (NODEL, NSIZE2)
      CALL IZERO1 (NPP, NP)
      CALL IZERO1 (NN, NSIZE2)
      CALL IZERO1 (IMAT, NE)
      CALL IZERO1 (NFIX, NP)
      CALL IZERO1 (NSTEPT, NSTEPP)
      CALL IZERO1 (NELF, NELSS1)
      CALL IZERO1 (NLI,   NIL1)
      CALL IZERO1 (NLT,   NLO1)
!-----------------------------------------------------------------
      CALL IZERO1 (NODLAY, NP)
      CALL IZERO1 (NELLAY, NE)
      CALL IZERO1 (NEQLAY, NEQNS)
!-----------------------------------------------------------------
!     CONSTRUCTION LAYER FOR EVERY TIME STEP
!-----------------------------------------------------------------
      CALL IZERO1 (KCONST, NSTEPS)
!-----------------------------------------------------------------
!     POST PROCESSING GRAPHIC FOR GID
!-----------------------------------------------------------------
      CALL IZERO1 (NODDUP, NP)
      CALL IZERO1 (NODNUM, NP)
!-----------------------------------------------------------------
!     2. TWO DIMENSIONAL INTEGER ARRAYS
!-----------------------------------------------------------------
      CALL IZERO2 (NOP,    NE, NCN)
      CALL IZERO2 (NODBON,NP, NDF)
      CALL IZERO2 (LPOIN, NP, NDF)
      CALL IZERO2 (IFPRE, NP, NDF)
      CALL IZERO2 (IFORG, NP, NDF)
      CALL IZERO2 (JFPRE, NP, NDF)
      CALL IZERO2 (LEQNS, NE, NSIZE)
      CALL IZERO2 (ISMOOT, NSIZE2, NSIZE2)
!-----------------------------------------------------------------
!     3. ONE DIMENSIONAL REAL ARRAYS
!-----------------------------------------------------------------
      CALL RZERO1 (R1, NEQNS)
      CALL RZERO1 (U, NB1)
      CALL RZERO1 (BIT4Q, NP)
      CALL RZERO1 (R2, NEQNS)
```

```
      CALL RZERO1 (ABDUM, 4)
      CALL RZERO1 (STM1, IIG)
      CALL RZERO1 (STM2, IIG)
      CALL RZERO1 (QHLOAD, LSIZE2)
      CALL RZERO1 (STLOAD, NSIZE1)
      CALL RZERO1 (PLOAD, NSIZE1)
      CALL RZERO1 (XG, NGAUS)
      CALL RZERO1 (QLOAD, NSIZE2)
      CALL RZERO1 (ELPORE, NSIZE2)
      CALL RZERO1 (SLOAD, NSIZE1)
      CALL RZERO1 (DARDUM, NGNG)
      CALL RZERO1 (P, NSIZE2)
      CALL RZERO1 (STRESS, ICS1B)
      CALL RZERO1 (CG, NGAUS)
      CALL RZERO1 (STRES1, ICS1B)
      CALL RZERO1 (W, NSIZE2)
      CALL RZERO1 (STRINC, ICS1B)
      CALL RZERO1 (SK, MTN73)
      CALL RZERO1 (SL, MTNSL)
!*****************************************************************
      CALL RZERO1 (ALOAD, 4)
      CALL RZERO1 (ABOUN, 4)
!*****************************************************************
      CALL RZERO1 (FIXED, NEQNS)
      CALL RZERO1 (FIXEB, NEQNS)
      CALL RZERO1 (FIXEC, NEQNS)
      CALL RZERO1 (STMOV, IN)
      CALL RZERO1 (TINCR, NSTEPS)
      CALL RZERO1 (ELF,    NELSS1)
      CALL RZERO1 (ELHF,   NELSS1)
      CALL RZERO1 (ELTEMP, NSIZE2)
      CALL RZERO1 (RLEVEL, NLEV1)
      CALL RZERO1 (DENSL,  NLEV1)
      CALL RZERO1 (STDUM,  ICS1B)
      CALL RZERO1 (DELR,   NRAD1)
      CALL RZERO1 (DELT,   NTET1)
      CALL RZERO1 (CORD2,  NROW)
      CALL RZERO1 (TFAC,   NSTEP1)
      CALL RZERO1 (TFACEL, NSTEP2)
      CALL RZERO1 (STARTV, ICS1B)
      CALL RZERO1 (APREDI, NP)
      CALL RZERO1 (XCORA,  NP)
      CALL RZERO1 (YCORA,  NP)
!-----------------------------------------------------------------
!     4. TWO DIMENSIONAL REAL ARRAYS
!-----------------------------------------------------------------
      CALL RZERO2 (PROP, NMAT,  I18)
      CALL RZERO2 (CORD, NP,     2)
      CALL RZERO2 (SMAT, NSIZE, NSIZE)
      CALL RZERO2 (COC, NSIZE1, NSIZE2)
      CALL RZERO2 (FLH, NSIZE2, NSIZE2)
      CALL RZERO2 (FLG, NSIZE2, NSIZE2)
      CALL RZERO2 (COM, NSIZE2, NSIZE1)
      CALL RZERO2 (PLK, NSIZE1, NSIZE1)
      CALL RZERO2 (TLH, LSIZE2, LSIZE2)
      CALL RZERO2 (COTU, LSIZE1, LSIZE2)
      CALL RZERO2 (TLL, LSIZE2, LSIZE2)
      CALL RZERO2 (TLG, LSIZE2, LSIZE2)
      CALL RZERO2 (TSL, LSIZE2, LSIZE2)
      CALL RZERO2 (VEL, 2,       NSIZE2)
!-----------------------------------------------------------------
      CALL RZERO2 (RILOAD, NIL1, NDF11)
      CALL RZERO2 (RLOAD, NLO1, NDFA)
```

```
      CALL RZERO2 (RGRAV, NP, 2)
      CALL RZERO2 (PFIXED, NP, NDF)
!-----------------------------------------------------------------------
      CALL RZERO2 (B, ICS1A, NSIZE1)
      CALL RZERO2 (D, ICS1A, ICS1A)
      CALL RZERO2 (DB, ICS1A, NSIZE1)
      CALL RZERO2 (BD, NSIZE1, ICS1A)
      CALL RZERO2 (CEL, 2, NSIZE2)
      CALL RZERO2 (DEL, 2, NSIZE2)
      CALL RZERO2 (CW, 2, NSIZE2)
      CALL RZERO2 (DW, 2, NSIZE2)
      CALL RZERO2 (PDUM, NSIZE2, NGNG)
      CALL RZERO2 (DIS, NDF, NP)
      CALL RZERO2 (STRAIN, NGNG, ICS1B)
      CALL RZERO2 (ST, NGNG, 6)
      CALL RZERO2 (BVALUE, NBN, NDF)
      CALL RZERO2 (FIXVAL, NP, NDF)
!-----------------------------------------------------------------------
!     POST PROCESSING GRAPHIC FOR GID
!-----------------------------------------------------------------------
      CALL RZERO2 (STNOP, NP, 4)
      CALL RZERO2 (SNNOP, NP, 4)
      CALL RZERO2 (SSTRNP, 4,  8)
      CALL RZERO2 (SSTNNP, 4,  8)
      CALL RZERO2 (SSTRGP,4,  4)
      CALL RZERO2 (SSTNGP,4,  4)
      CALL RZERO2 (TRANS, 4,  4)
!-----------------------------------------------------------------------
!     5. THREE DIMENSIONAL REAL ARRAYS
!-----------------------------------------------------------------------
      CALL RZERO3 (GASHT, IGS, MDF1, NP)
      CALL RZERO3 (DBDUM, ICS1A, NSIZE1, NGNG)
      CALL RZERO3 (BDUM, ICS1A, NSIZE1, NGNG)
      CALL RZERO3 (DEN,  2,       NGNG,  NEA1)
!***********************************************************************
      CALL GDATA  &
      (PROP,   TINCR,  CORD,   NOP,    IMAT,   NELM,   DELR,      &
       DELT,   CORD2,  GASHT,  RLEVEL, DENSL,  NLI,    RILOAD,    &
       NLT,    RLOAD,  NSTEPT, TFAC,   NELF,   ELF,    ELHF,      &
       TFACEL, STDUM,  STARTV, STMOV,  NELSS1, NELSS2, BB,        &
       MP,     ME,     MMAT,   MSTEPS, MDF,    I18,    NE1,       &
       NRAD1,  NTET1,  NLEV1,  NIL1,   NDF11,  NLO1,   NDFA,      &
       NSTEP1, NSTEP2, NELFL1, NELFL2, JN,     ICS1B,  NROW,      &
       MBN,    MTOTB,  IGS,    MDF1,   NSTEPP, NEQNS,  NPOIN,     &
       NDOFN,  BVALUE, NFIX,   NBC,    IFORG,  JFPRE,  FIXED,     &
       FIXEB,  FIXEC,  LCOM,   IGID,   KCONST, APREDI, MLAY,      &
       LEMBNK, NODBON, FIXVAL, IFGID)
!***********************************************************************
!      2005-03-26
!       ALLOCATE SWEIG (MLAY,NE,NSIZE1)
!***********************************************************************
      ALLOCATE (SWEIG(MLAY,ME, NSIZE1))
!***********************************************************************
!      2005-03-26
!     CLEAR TO ZERO OF SWEIG
!***********************************************************************
      CALL RZERO3 (SWEIG, MLAY, ME, NSIZE1)
!***********************************************************************
      MLEV1 = NLEV1
!***********************************************************************
      IF (NB > NBN) NB1 = NB * NDF
      MB1 = NB1
!***********************************************************************
```

```
      CALL SET (NOP, IONARY, MP, ME,   APREDI)
!*********************************************************************
      MARRAY = NARRAY * 2
!*********************************************************************
      IF (ICALC == 1) GO TO 15
      MTOT1B = 2 * NE * (JIG +  KSIZE1)
!*********************************************************************
      IF (MTOT1B > MTOTB) CALL ERROR2 (MTOT1B, MTOTB)
!*********************************************************************
      WRITE (IOUT, 101) MTOT1B
!---------------------------------------------------------------------
   15 CONTINUE
!---------------------------------------------------------------------
      IF (ISTOP == 1) STOP
!*********************************************************************
      CALL LOADSM (ISMOOT, KSIZES)
!*********************************************************************
      TIMA = 0.D0
      ISTEP = 1
!*********************************************************************
! *** CALL MAIN SUBROUTINES AND PRINT GASHT(I,J,K)
!---------------------------------------------------------------------
!     CLEAR AYYAYS FOR CPU TIEM CALCULATION TO ZERO
!---------------------------------------------------------------------
      DO KSTEP = 1, 180
        TSTEPS (KSTEP) = 0.D0
        TCPUTS (KSTEP) = 0.D0
        TDISKS (KSTEP) = 0.D0
        ST1S (KSTEP)   = 0.D0
        ST2S (KSTEP)   = 0.D0
        ST3S (KSTEP)   = 0.D0
        ST4S (KSTEP)   = 0.D0
        ST5S (KSTEP)   = 0.D0
        ST6S (KSTEP)   = 0.D0
        ST7S (KSTEP)   = 0.D0
        ST8S (KSTEP)   = 0.D0
        ST9S (KSTEP)   = 0.D0
        ST10S (KSTEP)  = 0.D0
        ST11S (KSTEP)  = 0.D0
        ST12S (KSTEP)  = 0.D0
        ST13S (KSTEP)  = 0.D0
      END DO
!---------------------------------------------------------------------
      ITPLOT = 1
      IGS1   = 1
      IF (NINSCH == 3) IGS1 = 3
      J18 = I18
      IF (NSOL == 1) GO TO 4201
!---------------------------------------------------------------------
!     USED BY PROFILE SOLVER
!---------------------------------------------------------------------
      NBLOCK = 1
!---------------------------------------------------------------------
      KBOUN  = 1
      CALL LINKIN  &
      (IFPRE,  MAXAJ,  LEQNS,  MHIGH,  NOP,     IFFIX,  NPOIN,     &
       NNODE,  NELEM,  NEVAB,  NDOFN,  NEQNS,   NEQN1,  NWK,       &
       KBOUN,  IOUT,   ISTAG,  NSTAG,  NTHERM,  JFPRE,  JFFIX,     &
       IBINT,  IMAT,   NODLAY, NELLAY, NEQLAY,  NODBON)
!---------------------------------------------------------------------
      IF (ISYMM == 1) NWBSU = NWK * 2
      IF (ISYMM /= 1) NWBSU = NWK * 4
!---------------------------------------------------------------------
```

```
!     WRITE GLOBAL STIFFNESS MATRIX ARRAY SIZE
!-----------------------------------------------------------------------
      WRITE (IOUT, 2331) NWK
 2331 FORMAT(//,70('-'),/, &
       2X,'***** GLOBAL STIFFNESS MATRIX SIZE *****',I10)
!-----------------------------------------------------------------------
!     USING PROFILE SOLUTION ARRAYS
!-----------------------------------------------------------------------
      KTMIN = NDF * NP
      IF (KTMIN < (2 * NSIZE2) ) KTMIN = 2 * NSIZE2
!-----------------------------------------------------------------------
!     IF YOU USE LPI OR LAHEY FORTRAN YOU MUST FIT    &
!     LTOTS AND LTOTU=4000
!     AU(M90)   BU(M91)   AL(M92)   BL(M93)
!***********************************************************************
!     DIVIDE MANY BLOCKS GLOBAL STIFFNESS MATRIX
!***********************************************************************
!CCC   WRITE (     *, 7654)
!7654 FORMAT (2X,'NUMBER OF BLOCK(MDIVID)')
!CCC   READ  (     *,     *)  MDIVID
!CCC   LGTOT = NWK/MDIVID +  NRHS
!***********************************************************************
      MTFU = MGFU
      IF (ISYMM == 1) MTFL = 1
      IF (ISYMM /= 1) MTFL = MTFU
!***********************************************************************
!     WRITE NWK, MTFU, AND MGFL
!***********************************************************************
!CCC   PRINT *,   'MDIVID   =', MDIVID
      PRINT *, 'NWK       =', NWK
      PRINT *, 'MTFU      =', MTFU
      PRINT *, 'MTFL      =', MTFL
      WRITE (IOUT, 5403) NWK, MTFU, MTFL
 5403 FORMAT (/,2X,'NWK   =',I10,/,2X,'MTFU =',I10,/,2X,'MTFL =',I10)
!***********************************************************************
!                                       STIFU(MTFU)
!                                       STIFFU(MTFU)
!                                       STIFL(MTFL)
!                                       STIFFL(MTFL)
!                                       DIS(NDF*NP)
!                                       VEL(2*NSIZE2)
!-----------------------------------------------------------------------
      ALLOCATE (STIFU(MTFU))
      ALLOCATE (STIFL(MTFL))
      ALLOCATE (STIFFU(MTFU))
      ALLOCATE (STIFFL(MTFL))
      ALLOCATE (DIAGN(NEQNS))
!-----------------------------------------------------------------------
      IF (NINSCH == 3) TIMA = TIMA +  TINCR (1)
!-----------------------------------------------------------------------
 4201 CONTINUE
!***********************************************************************
!     2005-03-22 TUESDAY
!     GAUSS INTEGRATION FACTORS
!***********************************************************************
      CALL GAUSSQ   &
       (MGAUS,   NGAUS,   CG,      XG)
!-----------------------------------------------------------------------
!     2005-03-23 WEDNSDAY
!     CALCULATE SELF LOAD (GRAVITY LOAD) FOR EMBANKMENT
!-----------------------------------------------------------------------
        IF (IBANK == 1)   &
        CALL GLGRAV   &
```

```
        (CORD,   NOP,    IMAT,   ME,     MMAT,   MP,     NSIZE,  &
         NSIZE1, NSIZE2, XG,     PROP,   SWEIG,  CG,     I18,    &
         MGAUS,  P,      W,      DEL,    DW,     CEL,    R1,     &
         MRHS,   LEQNS,  KLAY,   NELLAY, MLAY,   LEMBNK, RLOAD,  &
         NLO1,   NDFA)
!-----------------------------------------------------------------------
!     IF(NINSCH .EQ. 3)  TIMA = TIMA + TINCR(1)
!-----------------------------------------------------------------------
      NTOP = NINSCH
      TFR1 = TFRAC
!***********************************************************************
!     TITLE OF POST PROCESSING FOR GID
!***********************************************************************
      WRITE (IOGID, 1101)
 1101 FORMAT('GID POST RESULT FILE 1.0')
!***********************************************************************
!     START TIME STEPS
!-----------------------------------------------------------------------
!     CLEAR TTDIS(MP, MDF) TO ZERO    (22 FEB. 2004)
!-----------------------------------------------------------------------
      DO N = 1, NP
        DO I = 1, NDF
          TTDIS (N, I) = 0.0
        END DO
      END DO
!#######################################################################
      DO  MSTEP = 1, NSTEPS
!#######################################################################
        PRINT * , 'MSTEP =', MSTEP
!-----------------------------------------------------------------------
        CALL STCND (STMAN, LCOM)
!-----------------------------------------------------------------------
!     10 AUGUST, 2003 (10:17)
!     CONSTRUCTION LAYER FOR EVERY TIME STEP
!-----------------------------------------------------------------------
        KLAY = KCONST(MSTEP)
        IF (NTOP == 3.AND.MSTEP < 3) NINSCH = 2
        IF (NTOP == 3.AND.MSTEP < 3) TFRAC = 0.5D0
        NSTEP = MSTEP
        TIN = TINCR (MSTEP)
        ITER = 1
        IF (ISTEP == 0) GO TO 42
        IF (NSTEP == 1) GO TO 40
        IF (ISTEP >= NSTEP) GO TO 14
!-----------------------------------------------------------------------
!     ICONTOL VARIABLES FOR CHANGE OF BOUNDARY CONDITIONS
!-----------------------------------------------------------------------
        READ (IIN, 8888) DATCK
        WRITE ( *, 8888) DATCK
        READ  (IIN, * ) ISTEP, IBO
        WRITE (IOUT, 2) ISTEP, IBO
   14   IF (ISTEP /= NSTEP) GO TO 40
!-----------------------------------------------------------------------
        CALL BONCON  &
       (BVALUE, NFIX,   NBC,     ABDUM,  MP,     MDF,    MBN,     &
        NEQNS,  NPOIN,  NDOFN,   IFORG,  JFPRE,  FIXED,  FIXEB,   &
        FIXEC,  LCOM,   IGID,    NODBON, FIXVAL, IFGID)
!-----------------------------------------------------------------------
   40   IF (IOUTAP == 0) GO TO 42
        IF (NSTEP > 1) GO TO 42
!***********************************************************************
        CALL STCND (TSTART, LCOM)
!***********************************************************************
```

```
      WRITE (ITP5) NE, ( (NOP (N, M), M = 1, NCN), N = 1, NE)
      WRITE (ITP5) NE, (IMAT (N), N = 1, NE)
      WRITE (ITP5) NP, ( (CORD (N, M), M = 1, 2), N = 1, NP)
!*****************************************************************
      CALL STCND (TEND, LCOM)
      ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*****************************************************************
      DO I = 1, 3
        NDUM (I) = 0
      END DO
!*****************************************************************
      CALL STCND (TSTART, LCOM)
!*****************************************************************
      WRITE (ITP5) NB, (NDUM (I), I = 1, 3)
      IF (NB > 0) WRITE (ITP5) (NBC (I), NFIX (I),    &
          (BVALUE (I, J), J = 1, NDF), I = 1, NB)
!*****************************************************************
      CALL STCND (TEND, LCOM)
      ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*****************************************************************
      N = - 1
      IF (NINSCH /= 3) GO TO 4443
!*****************************************************************
      CALL STCND (TSTART, LCOM)
!*****************************************************************
      WRITE (ITP5) N, NP, ( (GASHT (1, J, K), GASHT (2, J, K), &
          GASHT (3, J, K), J = 1, NDF), K = 1, NP)
!*****************************************************************
      CALL STCND (TEND, LCOM)
      ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*****************************************************************
      GO TO 42
 4443   CONTINUE
!*****************************************************************
      CALL STCND (TSTART, LCOM)
!*****************************************************************
      WRITE (ITP5)   &
       N, NP, ( (GASHT (1, J, K), J = 1, NDF), K = 1, NP)
!*****************************************************************
      CALL STCND (TEND, LCOM)
      ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*****************************************************************
   42   CONTINUE
!*****************************************************************
!     26 DECEMBER, 2004
!     REVICE BOUNDARY CONDITIONS FOR INTERIA PORE WATER PRESSUER
!     BOUNDARY CONDITION VALUES
!     TO SET GROUND WATER LEVEL PRESSURE - ATMOSPHERIC PRESSURE
!     AND AFTER FINISHING PUMPAGE TO SET GROUND WATER LEVEL PRESSURE
!     ATMOSPHERIC PRESSURE = 101.325 KILO PASCAL
!     WATWR PRESSURE = 10.0 KILO-GRAM/SQURE-METER
!                    = 9.80665 KILO-PASCAL
!     PASCAL = N/SQURE-METER
!*****************************************************************
!     NFPUMP = TIME STEP STARTING PUMPAGE
!     NLPUMP = TIME STEP STOPPING PUMPAGE
!     (12 FEBRUARY, 2005)
!*****************************************************************
      IF (MSTEP < NFPUMP .OR. MSTEP > NLPUMP)  THEN
        CALL RBACON    &
       (IFPRE,  JFPRE,  IFFIX,  JFFIX,  KFFIX,  MP,     ME,     &
        MDF,    NSIZE,  FIXED,  FIXEB,  MRHS,   NPOIN,  NODLAY, &
        NELLAY, IMAT,   CORD,   NBC,    MBN,    BVALUE, NODBON, &
```

```
        FIXVAL, PUFIX,  NEQNS,  XCORA,  YCORA,  R1,     MSTEP)
      END IF
!*****************************************************************
      IF (MSTEP >= NFPUMP .OR. MSTEP <= NLPUMP)  THEN
        CALL DUBCON   &
       (IFPRE,  JFPRE,  IFFIX,  JFFIX,  KFFIX,  MP,     ME,     &
        MDF,    NSIZE,  FIXED,  FIXEB,  MRHS,   NPOIN,  NODLAY, &
        NELLAY, IMAT,   CORD,   NBC,    MBN,    BVALUE, NODBON, &
        FIXVAL, PUFIX,  NEQNS,  XCORA,  YCORA,  R1,     MSTEP)
      END IF
!*****************************************************************
      IF (NSOL == 1) GO TO 4203
!-----------------------------------------------------------------
!     USING BY PROFILE SOLVER
!     DEFINE BLOCK SIZE FOR SYMMETRIC OR UNSYMMETRIC SOLUTION
!-----------------------------------------------------------------
      IF (NSTEP > 1) GO TO 51
!-----------------------------------------------------------------
      CALL SBLOCK  &
     (MAXAJ,  ICOPL,  NCOLBV, NEQNS,   NEQN1,  MBLOCK, MTFU,    &
      NWK,    IOUT,   NBLOCK)
!-----------------------------------------------------------------
      IF (ISYMM == 1) KSREU = 8 * MTFU
      IF (ISYMM /= 1) KSREU = 8 * (MTFU + MTFL)
!-----------------------------------------------------------------
      OPEN (UNIT = ITP12, FILE = 'fi22.rplas',  &
       FORM = 'UNFORMATTED', STATUS = 'UNKNOWN',  &
       ACCESS = 'DIRECT', RECL = KSREU)
      OPEN (UNIT = ITP13, FILE = 'fi23.rplas',  &
       FORM = 'UNFORMATTED', STATUS = 'UNKNOWN',  &
       ACCESS = 'DIRECT', RECL = KSREU)
!-----------------------------------------------------------------
   51 CONTINUE
!-----------------------------------------------------------------
 4203 CONTINUE
!-----------------------------------------------------------------
  310 CONTINUE
!-----------------------------------------------------------------
      NLEV1 = NLEVEL
!-----------------------------------------------------------------
      CALL GSTIFM  &
     (IMAT,   IONARY, BB,     MTOTB,  MSTEPS, CORD,   NOP,      &
      GASHT,  NELF,   ELF,    ELHF,   STM1,   SMAT,   COC,      &
      FLH,    FLG,    COM,    PLK,    COTU,   TLH,    TLL,      &
      TLG,    TSL,    ELTEMP, QHLOAD, NODEL,  QLOAD,  ELPORE,   &
      SLOAD,  STLOAD, PLOAD,  B,      DB,     CEL,    DBDUM,    &
      BDUM,   DARDUM, PDUM,   P,      DEL,    STRESS, XG,       &
      CG,     PROP,   MP,     ME,     MDF,    NELFL1, NELFL2,   &
      IIG,    NSIZE,  NSIZE1, NSIZE2, LSIZE1, LSIZE2, ICS1A,    &
      ICS1B,  MGNG,   MGAUS,  NLEV1,  I18,    MMAT,   IGS,      &
      MDF1,   VEL,    NELSS1, NELSS2, W,      DW,     DEN,      &
      RLEVEL, DENSL,  ST,     STDUM,  STRINC, BD,     CW,       &
      STM2,   D,      STRES1, LCOM,   NEA1,   APREDI)
!-----------------------------------------------------------------
      ISTR1 = 0
!-----------------------------------------------------------------
!     2005-04-22 FRIDAY
!     ADD EMBANKMENT SELF-WEIGHT BY CONSTRUCTION LAYER (KLAY)
!-----------------------------------------------------------------
      IF (IBANK == 0)  NLOA = 1
      IF (IBANK == 1)  NLOA = NP
      IF (KLAY >= LEMBNK)  &
        CALL REVLOAD  &
```

```
          (NOP,    ME,     MP,     NSIZE,  NSIZE1, NSIZE2, SWEIG,  &
           NELLAY, MRHS,   KLAY,   RGRAV,  MLAY,   LEMBNK, TFAC,   &
           MSTEP,  MSTEPS, RLOAD,  KCONST)
!-----------------------------------------------------------------------
        CALL GLOAD  &
       (CORD,   NOP,    GASHT,  NLI,    RILOAD, NLT,    RLOAD,    &
        TFAC,   TFACEL, R1,     BIT4Q,  SMAT,   COC,    FLH,      &
        FLG,    COM,    COTU,   TLH,    TLL,    TLG,    TSL,      &
        ELTEMP, QHLOAD, NODEL,  QLOAD,  ELPORE, SLOAD,  STLOAD,   &
        MP,     ME,     NIL1,   NDF11,  NLO1,   NLOA,   NSTEP1,   &
        MRHS,   NSIZE,  NSIZE1, NSIZE2, LSIZE1, LSIZE2, NDFA,    &
        NSTEP2, IGS,    MDF1,   R2,     NEQNS,  LCOM,   APREDI,   &
        RGRAV,  IBANK)
!-----------------------------------------------------------------------
!     WRITE FOR CHECKING LOAD VECTOR R1(NEQNS)
!-----------------------------------------------------------------------
!        WRITE (KCHK1, 1110)
 1110 FORMAT(//, &
       2X,'*** AT MASTER LOAD VECROR R1 ***',/, &
       5X,'NODE',20X,'LOAD VECTOR R1',/)
        KD = 0
        DO  I = 1, NP
          DO  J = 1, NDF
            KD = KD+ 1
            ABDUM (J) = R1 (KD)
          END DO
!           IF (NTHERM.EQ.1)   &
!           WRITE (KCHK1, 1400) I, (ABDUM (J), J = 1, NDF)
!           IF (NTHERM.EQ.0)   &
!           WRITE (KCHK1, 1403) I, (ABDUM (J), J = 1, NDF)
 1400 FORMAT(2X,I9,5X,4(E15.6,1X))
 1403 FORMAT(2X,I9,5X,3(E15.6,1X))
        END DO
!-----------------------------------------------------------------------
!     THESE ARE NEEDED BY FRONTAL SOLVER
!-----------------------------------------------------------------------
        IF (NSOL > 1) GO TO 104
!-----------------------------------------------------------------------
        IF (NB <= 0) GO TO 104
        IBM = 0
        DO  I = 1, NB
          DO  J = 1, NDF
            IBM = IBM +  1
            U (IBM) = BVALUE (I, J)
            IF (NINSCH < 1) CYCLE
            IF (ITER /= NITER) GO TO 135
            IF (J /= NDF) U (IBM) = 0.D0
            CYCLE
  135       IF (J == NDF) U (IBM) = 0.D0
          END DO
        END DO
!*********************************************************************
!     CHECK OF U-MATRIX
!*********************************************************************
        WRITE (KCHK3, 3001) NB, NDF
        IBM = 0
        DO IA = 1, NB
          DO JA = 1, NDF
            IBM = IBM +  1
            ABOUN (JA) = U (IBM)
          END DO
          IF (NTHERM == 0)  &
            WRITE (KCHK3, 3003) IA, (ABOUN (KK), KK = 1, NDF)
```

```
          IF (NTHERM == 1)  &
            WRITE (KCHK3, 3005) IA, (ABOUN (KK), KK = 1, NDF)
        END DO
 3001 FORMAT(/, &
       2X,'BOUNDARY VALUE AT MASTER',/, &
       2X,'NB = ',I9,1X,'NDF =',I5,/, &
       2X,'NODE',1X,'BOUNDARY VALUES',/)
 3003 FORMAT(2X,I9,3(1X,E10.3))
 3005 FORMAT(2X,I9,4(1X,E10.3))
!---------------------------------------------------------------------
  104   CONTINUE
        IF (NSOL == 1) GO TO 4305
!*********************************************************************
!     2005-03-27 SUNDAY.
!     THIS IS NEEDED BY PROFILE SOLVER WITH CHANGING OF
!     BOUNDARY CONDITION AND ITS VALUES
!*********************************************************************
!     NFPUMP = TIME STEP STARTING PUMPAGE
!     NLPUMP = TIME STEP STOPPING PUMPAGE
!*********************************************************************
        CALL USEBOUN   &
       (IFFIX,  JFFIX,  KFFIX,  FIXED,  FIXEB,  FIXEC, R1,     &
        MRHS,   NFPUMP, NLPUMP, MSTEP,  IOUT,   LPOIN,  PFIXED, &
        NP,     NDF,    KCHEK)
!---------------------------------------------------------------------
!     2005-03-27 SUNDAY
!     FOR PROFILE SOLVER USING SEPERATE BOUNDARY VALUE SUBROUTINE
!     DIRICH
!---------------------------------------------------------------------
        KD = 0
        DO IP = 1, NP
          DO JD = 1, NDF
            KD = KD + 1
            IF (IFPRE (IP, JD) < 0)  R1 (KD) = FIXED (KD)
          END DO
        END DO
!*********************************************************************
!     CHECK OF R1-MATRIX
!*********************************************************************
      WRITE (KCHK3, 3201)  NRHS, NDF
      DO IP=1,NP
         IBM = (IP - 1)*NDF
         DO JA=1,NDF
            JBM = IBM + JA
            ALOAD(JA) = R1(JBM)
         END DO
      IF (NTHERM.EQ.0)  &
         WRITE (KCHK3, 3203)  IP, (ALOAD(KK),KK=1,NDF)
      IF (NTHERM.EQ.1)  &
         WRITE (KCHK3, 3205)  IP, (ALOAD(KK),KK=1,NDF)
      END DO
 3201 FORMAT(/,  &
        2X,'LOAD VECTOR VALUES AT MATER',/,  &
        2X,'NRHS = ',I5,1X,'NDF =',I5,/      &
        2X,'NODE',1X,'LOAD VECTOR VALUES',/)
 3203 FORMAT(2X,I5,3(1X,E10.3))
 3205 FORMAT(2X,I5,4(1X,E10.3))
!*********************************************************************
        CALL GASSEM  &
       (STIFU,  STIFFU, STIFL,  STIFFL, SMAT,   LEQNS,  MAXAJ,     &
        NCOLBV, NEVAB,  NELEM,  MBLOCK, MTFU,   MTFL,   NEQN1,     &
        KBOUN,  ISYMM,  ITP2,   ITP12,  ITP13,  NBLOCK, NSTEP,     &
        NDOFN,  LCOM)
```

```
!*****************************************************************
!      PROFILE SOLVER FOR THERMAL CONSOLIDATION
!      NTHERM = 1, AND NINSCH = 0
!      8TH MAY 2003 (18:14)
!*****************************************************************
       KBOUN =   1
       IDX   = - 1
       IDY   =   1
       ISW   = - 1
       IF (NSOL == 2) CALL DIRICH   &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     R1,     IFFIX,     &
        MAXAJ,  NCOLBV, NEQNS,  NEQN1,  MTFU,   MTFL,   MBLOCK,    &
        KBOUN,  IDX,    IDY,    ISW,    ISYMM, IOUT,    ITP12,     &
        NBLOCK, NSTEP,  ISTAG,  NTHERM, NITER, ITER,    LCOM,      &
        NEQLAY, KLAY,   KCONST, LEMBNK, MSTEPS)
!*****************************************************************
       IF (NSOL == 3) CALL RDIRICH   &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     R1,     IFFIX,     &
        MAXAJ,  NCOLBV, NEQNS,  NEQN1,  MTFU,   MTFL,   MBLOCK,    &
        KBOUN,  IDX,    IDY,    ISW,    ISYMM,  IOUT,   ITP12,     &
        NBLOCK, NSTEP,  ISTAG,  NTHERM, NITER,  ITER,   LCOM)
!*****************************************************************
       KBOUN = 1
       KKK   = 1
       IF (NSOL == 2) CALL UNSOL    &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     DIAGN, FIXED,     &
        FIXEB,  FIXEC,  GASHT,  DIS,    IFFIX,  MAXAJ, NCOLBV,    &
        NEQNS,  NEQN1,  MTFU,   MTFL,   MBLOCK, NPOIN, NDOFN,    &
        KBOUN,  KKK,    IOUT,   ITP12,  ITP13,  NBLOCK,IGS,       &
        MDF1,   ISTAG,  LCOM,   NODLAY, NEQLAY, KLAY,  LEMBNK)
!*****************************************************************
       IF (NSOL == 3) CALL BPSUS    &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     DIAGN,  FIXED,    &
        FIXEB,  IFFIX,  MAXAJ,  NCOLBV, NEQNS, NEQN1,  MTFU,      &
        MTFL,   MBLOCK, NPOIM,  NDOFN,  KBOUN, KKK,    IOUT,      &
        ITP12,  ITP13,  NBLOCK, IGS,    MDF1,  GASHT,  DIS,       &
        ISTAG,  LCOM)
!*****************************************************************
       KBOUN = 1
       KKK   = 2
       IF (NSOL == 2) CALL UNSOL    &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     DIAGN,  FIXED,    &
        FIXEB,  FIXEC,  GASHT,  DIS,    IFFIX, MAXAJ, NCOLBV,     &
        NEQNS,  NEQN1,  MTFU,   MTFL,   MBLOCK, NPOIN, NDOFN,    &
        KBOUN,  KKK,    IOUT,   ITP12,  ITP13,  NBLOCK,IGS,       &
        MDF1,   ISTAG,  LCOM,   NODLAY, NEQLAY, KLAY,  LEMBNK)
!*****************************************************************
       IF (NSOL == 3) CALL BPSUS    &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     DIAGN,  FIXED,    &
        FIXEB,  IFFIX,  MAXAJ,  NCOLBV, NEQNS, NEQN1,  MTFU,      &
        MTFL,   MBLOCK, NPOIN,  NDOFN,  KBOUN, KKK,    IOUT,      &
        ITP12,  ITP13,  NBLOCK, IGS,    MDF1,  GASHT,  DIS,       &
        ISTAG,  LCOM)
!*****************************************************************
       KBOUN = 1
       KKK   = 3
       IF (NSOL == 2) CALL UNSOL    &
       (STIFU,  STIFFU, STIFL,  STIFFL, R1,     DIAGN, FIXED,     &
        FIXEB,  FIXEC,  GASHT,  DIS,    IFFIX,  MAXAJ, NCOLBV,    &
        NEQNS,  NEQN1,  MTFU,   MTFL,   MBLOCK, NPOIN, NDOFN,    &
        KBOUN,  KKK,    IOUT,   ITP12,  ITP13,  NBLOCK,IGS,       &
        MDF1,   ISTAG,  LCOM,   NODLAY, NEQLAY, KLAY,  LEMBNK)
!*****************************************************************
```

```
      IF (NSOL == 3) CALL BPSUS    &
      (STIFU,  STIFFU, STIFL,  STIFFL, R1,    DIAGN,  FIXED,   &
       FIXEB,  IFFIX,  MAXAJ,  NCOLBV, NEQNS, NEQN1,  MTFU,    &
       MTFL,   MBLOCK, NPOIN,  NDOFN,  KBOUN, KKK,    IOUT,    &
       ITP12,  ITP13,  NBLOCK, IGS,    MDF1,  GASHT,  DIS,     &
       ISTAG,  LCOM)
!*****************************************************************
      KBOUN =   1
      IDX   =   0
      IDY   = - 1
      ISW   =   0
      IF (NSOL == 2.AND.IBINT /= 1) CALL DIRICH    &
      (STIFU,  STIFFU, STIFL,  STIFFL, R1,    R2,     IFFIX,  &
       MAXAJ,  NCOLBV, NEQNS,  NEQN1,  MTFU,  MTFL,   MBLOCK, &
       KBOUN,  IDX,    IDY,    ISW,    ISYMM, IOUT,   ITP12,  &
       NBLOCK, NSTEP,  ISTAG,  NTHERM, NITER, ITER,   LCOM,   &
       NEQLAY, KLAY,   KCONST, LEMBNK, MSTEPS)
      IF (NSOL == 2.AND.IBINT == 1) CALL DIRICH    &
      (STIFU,  STIFFU, STIFL,  STIFFL, R1,    R2,     IFFIX,  &
       MAXAJ,  NCOLBV, NEQNS,  NEQN1,  MTFU,  MTFL,   MBLOCK, &
       KBOUN,  IDX,    IDY,    ISW,    ISYMM, IOUT,   ITP12,  &
       NBLOCK, NSTEP,  ISTAG,  NTHERM, NITER, ITER,   LCOM,   &
       NEQLAY, KLAY,   KCONST, LEMBNK, MSTEPS)
!*****************************************************************
      IF (NSOL > 1) GO TO 4207
!-----------------------------------------------------------------
!     TO USE FRONTAL SOLVER
!-----------------------------------------------------------------
 4305 CONTINUE
!-----------------------------------------------------------------
      CALL FRSOL   &
     (NOP,    NFIX,   NBC,    R1,     SMAT,   SL,     NN,     &
      U,      SK,     MP,     ME,     MRHS,   LNBNDF, NSIZE2, &
      NSIZE,  MTN73,  LMTNSL, LCOM)
!-----------------------------------------------------------------
      CALL FRBAK   &
     (GASHT,  R1,     U,      DIS,    SK,     MP,     MDF,    &
      MRHS,   LNBNDF, IGS,    MDF1,   MTN73,  LCOM,   APREDI)
!-----------------------------------------------------------------
 4207 CONTINUE
!-----------------------------------------------------------------
      IF (NINSCH > 1.AND.ITER == NITER) GO TO 108
      CALL STRAIS  &
       (PROP,   CORD,   NOP,    IMAT,   GASHT,  IONARY, STM1,   &
        STM2,   DBDUM,  BDUM,   DARDUM, PDUM,   STRAIN, ST,     &
        BB,     MP,     ME,     MMAT,   I18,    IIG,    MGNG,   &
        NSIZE1, ICS1A,  ICS1B,  NSIZE2, MTOTB,  IGS,    MDF1,   &
        LCOM,   APREDI)
!*****************************************************************
  308 CONTINUE
      IF (NINSCH == 0) GO TO 108
      ITER = ITER + 1
!-----------------------------------------------------------------
      GO TO 310
!-----------------------------------------------------------------
  108 CONTINUE
!-----------------------------------------------------------------
      NADUM = NW
      NW    = NR
      NR    = NADUM
      NDF1  = NDF - 1
      IF (NTHERM == 1) NDF1 = NDF1 - 1
      IF (NINSCH < 1) GO TO 415
```

```
      IF (MSTEP == 2) NINSCH = NTOP
      IF (MSTEP == 2) TFRAC = TFR1
      DO  I = 1, NP
        DO  J = 1, NDF1
          GASHT (1, J, I) = GASHT (2, J, I)
          GASHT (2, J, I) = GASHT (3, J, I)
          IF (NINSCH == 2) GASHT (1, J, I) = GASHT (3, J, I)
        END DO
        IF (NTHERM == 1)  &
          GASHT (1, NDF - 1, I) = GASHT (2, NDF - 1, I)
        IF (NTHERM == 1)  &
          GASHT (2, NDF - 1, I) = GASHT (3, NDF - 1, I)
        IF (NTHERM == 1.AND.NINSCH == 2)  &
            GASHT (1, NDF - 1, I) = GASHT (3, NDF - 1, I)
        GASHT (1, NDF, I) = GASHT (2, NDF, I)
        GASHT (2, NDF, I) = GASHT (3, NDF, I)
      END DO
      GO TO 1053
  415  CONTINUE
      DO  I = 1, NP
        DO  J = 1, NDF1
          GASHT (1, J, I) = GASHT (2, J, I) +  GASHT (1, J, I)
        END DO
        IF (NTHERM == 1)  &
          GASHT (1, NDF - 1, I) = GASHT (2, NDF - 1, I)
        APREDI (I) = GASHT (1, NDF, I)
        GASHT (1, NDF, I) = GASHT (2, NDF, I)
      END DO
      GO TO 106
 1053  IF (NSTEP > 1) GO TO 103
!-----------------------------------------------------------------------
      IF (NSMITP > 0) CALL PRODSM  &
      (NOP,    GASHT,  ISMOOT, ME,     MP,     MDF1,   NSMITP, &
       3,      KSIZES, 3,      IGS)
!-----------------------------------------------------------------------
      IF (NSMITT > 0) CALL PRODSM  &
      (NOP,    GASHT,  ISMOOT, ME,     MP,     MDF1,   NSMITT, &
       4,      KSIZES, 3,      IGS)
!-----------------------------------------------------------------------
      GO TO 103
  106  IF (NSTEP > 1) GO TO 103
!-----------------------------------------------------------------------
      IF (NSMITP > 0) CALL PRODSM  &
      (NOP,    GASHT,  ISMOOT, ME,     MP,     MDF1,   NSMITP, &
       3,      KSIZES, 1,      IGS)
!-----------------------------------------------------------------------
      IF (NSMITT > 0) CALL PRODSM  &
      (NOP,    GASHT,  ISMOOT, ME,     MP,     MDF1,   NSMITT, &
       4,      KSIZES, 1,      IGS)
!-----------------------------------------------------------------------
  103  IF (NINSCH /= 3) GO TO 1054
      TIN = TIN / 2.D0
      IF (NINSCH == 3.AND.NSTEP == 2) TIN = 2.D0 * TIN
 1054  TIMA = TIMA +  TIN
!-----------------------------------------------------------------------
!      PRINT DISPLACEMENTS FOR ALL SOLVER
!-----------------------------------------------------------------------
      WRITE (IODIS, 110) TIMA, MSTEP
      WRITE (IODIS, 115)
      DO  N = 1, NP
        IF (NINSCH < 1) GO TO 663
!-----------------------------------------------------------------------
!     CALCULATE TOTAL DISPLACEMENTS FOR TIME STEPS (01 MARCH 2004)
```

```
!     (GASHT(1,IDF,IPOIN) =
!                   TOTAL DISPLACEMENTS UNTIL TIHIS TIME STEP
!---------------------------------------------------------------------
          DO I = 1, NDF
            DIS (I, N) = GASHT (1, I, N)
            TTDIS (N, I) =   DIS (I, N)
          END DO
!---------------------------------------------------------------------
          IF (NTHERM == 0)  &
            WRITE (IODIS, 112)    &
             N, (TTDIS (N, I), I = 1, NDF), NODLAY(N)
          IF (NTHERM == 1)  &
            WRITE (IODIS, 212)  &
             N, (TTDIS (N, I), I = 1, NDF), NODLAY(N)
          GO TO 662
!---------------------------------------------------------------------
  663      CONTINUE
!---------------------------------------------------------------------
          DO I = 1, NDF
            DIS (I, N) = GASHT (1, I, N)
            TTDIS (N, I) = DIS (I, N)
          END DO
!---------------------------------------------------------------------
          IF (NTHERM == 0)    THEN
            WRITE (IODIS, 112)    &
             N, (TTDIS (N, I), I = 1, NDF), NODLAY(N)
          END IF
          IF (NTHERM == 1)    THEN
            WRITE (IODIS, 212)    &
             N, (TTDIS (N, I), I = 1, NDF), NODLAY(N)
          END IF
  662    CONTINUE
        END DO
!---------------------------------------------------------------------
!     GID GRAPHICS DATA
!---------------------------------------------------------------------
        NGAU2   = NGAUS*NGAUS
        NSTRES = 4
!*********************************************************************
        CALL RESFLAVIA   &
       (TTDIS,   ST,      STRAIN, NPOIN,   NELEM,   NDOFN,   NGAU2,      &
        NSTRES, NTHERM, IOGID,   NGAUS,   MGNG,    NOP,     NODDUP,     &
        NODNUM, STNOP,   SNNOP,   SSTRNP, SSTNNP, SSTRGP, SSTNGP,     &
        NCN)
!*********************************************************************
!     PRINT GRAPHIC DATA
!---------------------------------------------------------------------
        IF (NTPLOT == 0) GO TO 18
        IF (NSTEPT (ITPLOT) /= NSTEP) GO TO 18
        ITPLOT = ITPLOT +  1
        READ (IIN, * ) NGRX, NGRY, NGRP, NGRT
        IF (NGRX == 0) GO TO 11
!---------------------------------------------------------------------
        DO I = 1, NGRX
          IGS2 = 1
          CALL GRAPHI (CORD, GASHT, NPP, NP, IGS, MDF1, IGS1, IGS2)
        END DO
!---------------------------------------------------------------------
   11   IF (NGRY == 0) GO TO 144
        DO I = 1, NGRY
          IGS2 = 2
          CALL GRAPHI (CORD, GASHT, NPP, NP, IGS, MDF1, IGS1, IGS2)
        END DO
```

```
!-----------------------------------------------------------------------
  144   IF (NGRP == 0) GO TO 4416
        DO I = 1, NGRP
          IGS2 = 3
          CALL GRAPHI (CORD, GASHT, NPP, NP, IGS, MDF1, IGS1, IGS2)
        END DO
!-----------------------------------------------------------------------
 4416   IF (NGRT == 0.OR.NTHERM == 0) GO TO 18
        DO I = 1, NGRT
          IGS2 = 4
          CALL GRAPHI (CORD, GASHT, NPP, NP, IGS, MDF1, IGS1, IGS2)
!-----------------------------------------------------------------------
        END DO
!-----------------------------------------------------------------------
!       PRINT REACTIONS FOR FRONTAL SOLVER
!-----------------------------------------------------------------------
   18   IF (MBN == 0) GO TO 680
        IF (IWRIT == 3) GO TO 680
        WRITE (IODIS, 113)
        IF (NSOL > 1) GO TO 731
        DO   N = 1, MBN
          M = NBC (N)
          LM = M * NDF
          LL = LM +  1 - NDF
          NFIXP = NFIX (N) - NFIX (N) / 100 * 100
          NFIXP = NFIXP - NFIXP / 10 * 10
          IF (NFIXP == 0) GO TO 674
          R1 (LM) = R1 (LM) / CURILL - BIT4Q (M)
  674     CONTINUE
          IF (NTHERM == 0)    &
            WRITE (IODIS, 112) M, (R1 (L), L = LL, LM), NODLAY(M)
          IF (NTHERM == 1)   &
            WRITE (IODIS, 212) M, (R1 (L), L = LL, LM), NODLAY(M)
        END DO
        IF (NSOL == 1) GO TO 734
!-----------------------------------------------------------------------
!       PRINT REACTIONS FOR PROFILE SOLVER
!-----------------------------------------------------------------------
  731   CONTINUE
        DO   N = 1, MBN
          M = NBC (N)
          LM = M * NDF
          LL = LM +  1 - NDF
          NFIXP = NFIX (N) - NFIX (N) / 100 * 100
          NFIXP = NFIXP - NFIXP / 10 * 10
          IF (NFIXP == 0) GO TO 774
          R2 (LM) = R2 (LM) / CURILL - BIT4Q (M)
  774     CONTINUE
          IF (NTHERM == 0)   &
            WRITE (IODIS, 112) M, (R2 (L), L = LL, LM), NODLAY(M)
          IF (NTHERM == 1)   &
            WRITE (IODIS, 212) M, (R2 (L), L = LL, LM), NODLAY(M)
        END DO
!-----------------------------------------------------------------------
  734   CONTINUE
  680   CONTINUE
!**********************************************************************
        IF (IOUTAP == 0) GO TO 3107
        IF (NINSCH /= 3) GO TO 123
!**********************************************************************
        CALL STCND (TSTART, LCOM)
!**********************************************************************
        WRITE (ITP5)   &
```

```
        NSTEP, NP, TIMA, ( (GASHT (1, J, K), GASHT (2, J, K),    &
        GASHT (3, J, K), J = 1, NDF), K = 1, NP)
!*********************************************************************
        CALL STCND (TEND, LCOM)
        ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*********************************************************************
        GO TO 124
  123   CONTINUE
!*********************************************************************
        CALL STCND (TSTART, LCOM)
!*********************************************************************
        WRITE (ITP5) NSTEP, NP, TIMA,    &
            ( (GASHT (1, J, K), J = 1, NDF), K = 1, NP)
!*********************************************************************
        CALL STCND (TEND, LCOM)
        ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*********************************************************************
  124   CONTINUE
        IF (NW == 2) GO TO 121
!*********************************************************************
        IF (ICALC == 1) CALL STCND (TSTART, LCOM)
!*********************************************************************
        IF (ICALC == 1) REWIND ITP6
        IF (ICALC == 1) READ (ITP6) (STMOV (J), J = 1, IN)
!*********************************************************************
        IF (ICALC == 1) CALL STCND (TEND, LCOM)
        IF (ICALC == 1) ST6S (NSTEP) = ST6S (NSTEP) + (TEND-TSTART)
!*********************************************************************
!---------------------------------------------------------------------
        IF (ICALC == 0) CALL MOVE1 (BB, STMOV, MTOTB, IN, 0, 0, IN)
!---------------------------------------------------------------------
        IF (ICALC == 1) REWIND ITP6
        GO TO 122
  121   CONTINUE
!*********************************************************************
        IF (ICALC == 1) CALL STCND (TSTART, LCOM)
!*********************************************************************
        IF (ICALC == 1) REWIND ITP7
        IF (ICALC == 1) READ (ITP7) (STMOV (J), J = 1, IN)
!*********************************************************************
        IF (ICALC == 1) CALL STCND (TEND, LCOM)
        IF (ICALC == 1) ST7S (NSTEP) = ST7S (NSTEP) + (TEND-TSTART)
!---------------------------------------------------------------------
        IF (ICALC == 0) CALL MOVE1 (BB, STMOV, MTOTB, IN, 0, IN, IN)
!---------------------------------------------------------------------
        IF (ICALC == 1) CALL STCND (TSTART, LCOM)
!*********************************************************************
        IF (ICALC == 1) REWIND ITP7
  122   CONTINUE
        IF (ICALC == 1) CALL STCND (TSTART, LCOM)
        WRITE (ITP5) (STMOV (J), J = 1, IN)
!*********************************************************************
        IF (ICALC == 1) CALL STCND (TEND, LCOM)
        IF (ICALC == 1) ST5S (NSTEP) = ST5S (NSTEP) + (TEND-TSTART)
!*********************************************************************
 3107   CONTINUE
        CALL STCND (STEND, LCOM)
!*********************************************************************
!       CALCULATE CPU TIME, DISK ACCESS TIME,    &
!       RUNTIME AT EVERY TIME STEP
!*********************************************************************
        TSTEPS (NSTEP) = STEND-STMAN
        PRINT *, "***RUN TIME AT THIS STEP***", TSTEPS (NSTEP)
```

```
        TDISKS (NSTEP) = ST1S (NSTEP) + ST2S (NSTEP) + ST3S (NSTEP) + &
            ST4S (NSTEP) + ST5S (NSTEP) + ST6S (NSTEP) +    &
            ST7S (NSTEP) + ST8S (NSTEP) + ST9S (NSTEP) +    &
            ST10S (NSTEP) + ST11S (NSTEP) + ST12S (NSTEP) + &
            ST13S (NSTEP)
        TCPUTS (NSTEP) = TSTEPS (NSTEP) - TDISKS (NSTEP)
!-----------------------------------------------------------------------
        PRINT 6000
        PRINT 6010, TSTEPS (NSTEP), TDISKS (NSTEP), TCPUTS (NSTEP)
        WRITE (IOUT, 6000)
        WRITE (IOUT, 6010)    &
         TSTEPS (NSTEP), TDISKS (NSTEP), TCPUTS (NSTEP)
 6000 FORMAT (/, &
       1X,' RUN TIME OF EVERY TIME STEP ',/, &
       1X,'TOTAL TIME',7X,'DISK TIME',8X,'CPU TIME')
 6010 FORMAT (1X,D16.8,1X,D16.8,1X,D16.8)
!#######################################################################
!      END OF TIME STEP (MSTEP)
!#######################################################################
      END DO
!#######################################################################
!      CALCULATION OF TOTAL, DISK ACCESS, AND PURE RUN TIME
!***********************************************************************
      RUNT = 0.D0
      DSKT = 0.D0
      DO   MMM = 1, NSTEPS
        RUNT = RUNT +  TSTEPS (MMM)
        DSKT = DSKT +  TDISKS (MMM)
        PRINT * , 'TDISKS(', MMM, ' )=', TDISKS (MMM)
        PRINT * , 'DSKT =', DSKT
      END DO
      CPTOT = RUNT - DSKT
!-----------------------------------------------------------------------
      WRITE (IOUT, 5601)
      WRITE (IOUT, 5603) RUNT, DSKT, CPTOT
 5601 FORMAT(//, &
       2X,'**** RUN TIME THIS PROGRAM ***',/)
 5603 FORMAT( /, &
       'TOTAL RUN',10X,'DISK ACCESS',7X,'PURE RUN(SEC)',/, &
        D16.8,2X,D16.8,2X,D16.8)
!-----------------------------------------------------------------------
!      FORMATS
!-----------------------------------------------------------------------
    2 FORMAT (//,' ISTEP = ',I5,/, &
       ' IBO    = ',I5,/)
  110 FORMAT (/,70('-'),/, &
       ' DISPLACEMENT PORE WATER PRESSURES AND TEMPERATURE ',/, &
       ' AT TIME = ',F15.5,2X,'ITERATION NO =',I8,/)
  112 FORMAT (I9 ,3(1X,E15.6),2X,I5)
  212 FORMAT (I9 ,4(1X,E15.6),2X,I5)
  113 FORMAT (//,' REACTIONS ',/)
  115 FORMAT (1X,   &
       'NODE NO.', &
       1X,'X-DISPLACEMENT', 3X,'Y-DISPLACEMENT', &
       4X,'PORE PRESSURE',4X,'TEMPERATURE',2X,'LAYER',/)
 8888 FORMAT (A80)
!-----------------------------------------------------------------------
!      FOR PROFILE SOLVER
!-----------------------------------------------------------------------
      IF (NSOL > 1) WRITE (IOUT, 99)
!-----------------------------------------------------------------------
      NCOUNT = NCOUNT +  1
      IF (NCOUNT <= NPROB) GO TO 10
```

```
      REWIND ITP4
      REWIND ITP5
!-----------------------------------------------------------------------
!     FILE CLOSE
!-----------------------------------------------------------------------
      CALL FCLOSE
!-----------------------------------------------------------------------
      CALL STCND (TMEND, LCOM)
      IF (NSOL == 1) WRITE (IOUT, 3010)
      IF (NSOL > 1) WRITE (IOUT, 3020)
!-----------------------------------------------------------------------
!     FORMATS
!-----------------------------------------------------------------------
    7 FORMAT (A72)
   99 FORMAT (//,70('-'),/)
  100 FORMAT (2X,'STORAGE OF VECTOR A   IS',1X,I10)
  101 FORMAT (2X,'STORAGE OF VECTOR BB IS',1X,I9)
 1001 FORMAT (//)
 1002 FORMAT (//,A72)
 3010 FORMAT (2X,'THIS WAS SOLVED BY FRONTAL METHOD')
 3020 FORMAT (2X,'THIS WAS SOLVED BY BLOCKED PROFILE METHOD')
      STOP
      END PROGRAM RPLASCON
```

Appendix 2

Subroutine DIRICH

```
SUBROUTINE DIRICH    &
       (STIFU,  STIFFU, STIFL,  STIFFL, XX,      YY,      IFFIX,      &
        MAXAJ,  NCOLBV, NEQNS,  NEQN1,  MTFU,    MTFL,    MBLOCK,     &
        KBOUN,  IDX,    IDY,    ISW,    ISYMM,   IOUT,    ITP12,      &
        NBLOCK, NSTEP,  ISTAG,  NTHERM, NITER,   ITER,    LCOM,       &
        NEQLAY, KLAY,   KCONST, LEMBNK, MSTEPS)
!***************************************************************
!     TO TREAT BOUNDARY CONDITION TO CALCULATE REACTIONS,,,,ETC.
!
!     MULTIFLY (AND ADDITION OR SUBTRACTION) MATRIX
!     IF  ISW=0    THEN DISPL(IEQNS)=STIFI(KK)*DISPL(IEQNS)
!     IF  ISW=1    THEN BALANCE CHECK OF SOLUTION (NOT USED)
!     IF  ISW=(-1)  THEN REACT(IEQNS)=REACT(IEQNS) - STIFI(KK)*
!                                                  DISPL(IEQNS)
!---------------------------------------------------------------
!         IDX<0    ONLY NEGATIVE INDEX COLUMNS ARE MULTIPLIED
!         IDX=0    ALL COLUMN ARE MULTIPLIED
!         IDX>0    ONLY POSITIVE INDEX COLUMN ARE MULTIPLIED
!---------------------------------------------------------------
!         IDY<0    ONLY NEGATIVE INDEX COLUMNS ARE OBTAINED
!         IDY=0    ALL COLUMN ARE MULTIPLIED
!         IDY>0    ONLY POSITIVE INDEX COLUMNS ARE OBTAINED
!---------------------------------------------------------------
!     12TH  FEBERUARY 2004
!     DEVELOPED BY  JAE-SUN  KANG
!     DEPARTMENT OF CIVIL ENGINEERING,      ENGINEERING COLLEGE
!     KYUNG  NAM  UNIVERSITY
!     MASAN,  KOREA (631-701)
!***************************************************************
      USE KCHECK_COM
      USE NSSOL_COM
      USE CTIMS_COM
      USE CTMAS_COM
!***************************************************************
      IMPLICIT DOUBLE PRECISION (A - H, O - Z)
!***************************************************************
      DIMENSION   &
       XX (NEQNS), YY (NEQNS), STIFU (MTFU), STIFL (MTFL), &
       STIFFU (MTFU), STIFFL (MTFL), IFFIX (NEQNS),          &
       MAXAJ (NEQN1), NCOLBV (MBLOCK)
      DIMENSION NEQLAY(NEQNS), ISTAG (NEQNS), KCONST (MSTEPS)
!***************************************************************
!     CHECK OF ARRAY
!***************************************************************
      WRITE (KCHK1, 7000)
 7000 FORMAT(/,1X,'*** AT DIRICH    ***')
!     WRITE (KCHK1, 7001) NEQNS, MTFU, MTFL, NEQN1, MBLOCK
 7001 FORMAT( &
       1X,'NEQNS  =',I5,1X,'MTFU   =',I5,1X,'MTFL   =',I5,/, &
       1X,'NEQN1  =',I5,1X,'MBLOCK =',I5)
!***************************************************************
      PRINT *, 'KLAY =', KLAY
!---------------------------------------------------------------
      NTAPE = ITP12
```

```
      KHBB = 0
      NEQNC = NEQNS + 1
!-----------------------------------------------------------------------
      DO  NJ = 1, NBLOCK
!***** DIRECT FILE *****
!***********************************************************************
        IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM == 1)  &
          READ (NTAPE, REC = NJ) (STIFU (IQ), IQ = 1, MTFU)
        IF (ISYMM == 1) CALL STCND (TEND, LCOM)
        ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!***********************************************************************
        IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM /= 1)  &
          READ (NTAPE, REC = NJ) (STIFU (IQ), IQ = 1, MTFU),  &
                                 (STIFL (IQ), IQ = 1, MTFL)
        IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
        ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!***********************************************************************
!***** DIRECT FILE *****
        NCOLB = NCOLBV (NJ)
        MM = MAXAJ (KHBB + 1) - 1
!-----------------------------------------------------------------------
        DO  N = 1, NCOLB
          YV = 0.D0
          IF (NSTAG == 1.AND.NTHERM == 1) THEN
            IF (ITER /= NITER.AND.ISTAG (KHBB + N) == 2) GO TO 720
            IF (ITER == NITER.AND.ISTAG (KHBB + N) == 1) GO TO 720
          END IF
!***********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!***********************************************************************
          IF (NEQLAY(KHBB+N) > KLAY)  GO TO 720
!***********************************************************************
          IF ( (IFFIX (KHBB + N) ) * IDY < 0) GO TO 720
!-----------------------------------------------------------------------
          KN = MAXAJ (KHBB + N) - MM
          KL = KN + 1
          KU = MAXAJ (KHBB + N + 1) - MM - 1
          KS = KHBB + N
          K = KS
          IF (KS <= NEQNC) GO TO 825
          K = NEQNC + 1
          KL = KL + KS - NEQNC - 1
  825     CONTINUE
          IF ((KU-KL) < 0)  GO TO 820
          IF ((KU-KL) >= 0)  GO TO 830
  830     C = 0.D0
!-----------------------------------------------------------------------
          DO  KK = KL, KU
            K = K - 1
            IF (NSTAG == 1.AND.NTHERM == 1) THEN
              IF (ITER /= NITER.AND.ISTAG (K) == 2) GO TO 840
              IF (ITER == NITER.AND.ISTAG (K) == 1) GO TO 840
            END IF
!***********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!***********************************************************************
            IF (NEQLAY(K) > KLAY)   GO TO 840
!***********************************************************************
            IF ( (IFFIX (K) ) * IDX < 0) GO TO 840
```

```
          IF (ISYMM == 1) C = C + STIFU (KK) * XX (K)
          IF (ISYMM /= 1) C = C + STIFL (KK) * XX (K)
  840     CONTINUE
        END DO
        YV = YV + C
  820   CONTINUE
        IF (N == NCOLB) GO TO 960
!-----------------------------------------------------------------------
        NP1 = N
        DO  J = NP1, NCOLB
          IF (NSTAG == 1.AND.NTHERM == 1) THEN
            IF (ITER /= NITER.AND.ISTAG (KHBB + J) == 2)  &
              GO TO 910
            IF (ITER == NITER.AND.ISTAG (KHBB + J) == 1)  &
              GO TO 910
          END IF
!***********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!***********************************************************************
          IF (NEQLAY(KHBB+ J) > KLAY)  GO TO 910
!***********************************************************************
          IF ( (IFFIX (KHBB + J) ) * IDX < 0) GO TO 910
          KN = MAXAJ (KHBB + J) - MM
          KU = MAXAJ (KHBB + J + 1) - MM - 1
          KH = KU - KN
          IF (KHBB + J == 1) GO TO 990
          JT = KHBB + J - KH
          IF ( (KHBB + N) < JT) GO TO 910
  990      KK = KN + (KHBB + J) - (KHBB + N)
          YV = YV + STIFU (KK) * XX (KHBB + J)
  910     CONTINUE
        END DO
  960   CONTINUE
        IF (NBLOCK == 1) GO TO 920
        IF (NJ == NBLOCK) GO TO 920
!-----------------------------------------------------------------------
!     CALCULATION OF FORWARD BLOCKS
!-----------------------------------------------------------------------
        KHB = KHBB + NCOLB
        NJ1 = NJ + 1
        DO  KB = NJ1, NBLOCK
          KMM = MAXAJ (KHB + 1) - 1
          KCOLB = NCOLBV (KB)
!***** DIRECT FILE *****
!***********************************************************************
          IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
          IF (ISYMM == 1)  &
            READ (NTAPE, REC = KB)   (STIFFU (IQ), IQ = 1, MTFU)
          IF (ISYMM == 1) CALL STCND (TEND, LCOM)
          IF (ISYMM == 1)  &
            ST12S (NSTEP) = ST12S (NSTEP) + (TEND- TSTART)
!***********************************************************************
          IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
          IF (ISYMM /= 1)  &
            READ (NTAPE, REC = KB) (STIFFU (IQ), IQ = 1, MTFU), &
                                   (STIFFL (IQ), IQ = 1, MTFL)
          IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
          IF (ISYMM /= 1)  &
            ST12S (NSTEP) = ST12S (NSTEP) + (TEND- TSTART)
!***********************************************************************
!***** DIRECT FILE *****
          DO  J = 1, KCOLB
```

```
!*****************************************************************
              IF (NSTAG == 1.AND.NTHERM == 1) THEN
                IF (ITER /= NITER.AND.ISTAG (KHB + J) == 2)   &
                  GO TO 9100
                IF (ITER == NITER.AND.ISTAG (KHB + J) == 1)   &
                  GO TO 9100
              END IF
!*****************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*****************************************************************
              IF (NEQLAY(KHB+ J) > KLAY)   GO TO 9100
!*****************************************************************
              IF ( (IFFIX (KHB + J) ) * IDX < 0) GO TO 9100
              KN = MAXAJ (KHB + J) - KMM
              KU = MAXAJ (KHB + J + 1) - KMM - 1
              KH = KU - KN
              JT = KHB + J - KH
              IF ( (KHBB + N) < JT) GO TO 9100
 9900          KK = KN + (KHB + J) - (KHBB + N)
              YV = YV + STIFFU (KK) * XX (KHB + J)
 9100          CONTINUE
            END DO
            KHB = KHB + KCOLB
          END DO
!-----------------------------------------------------------------
  920      CONTINUE
!-----------------------------------------------------------------
!CCCC IF(ISW) 7100,7200,7300
!-----------------------------------------------------------------
          IF (ISW < 0) YY (KHBB + N) = YY (KHBB + N) - YV
          IF (ISW == 0) YY (KHBB + N) = YV
          IF (ISW > 0) YY (KHBB + N) = YY (KHBB + N) + YV
!-----------------------------------------------------------------
  720      CONTINUE
        END DO
        KHBB = KHBB + NCOLB
      END DO
!-----------------------------------------------------------------
      RETURN
      END SUBROUTINE DIRICH
```

Appendix 3

Subroutine UNSOL

```
SUBROUTINE UNSOL  &
       (STIFU,  STIFFU, STIFL,  STIFFL, V,       DIAGN,  FIXED,      &
        FIXEB,  FIXEC,  GASHT,  DIS,     IFFIX,  MAXAJ,  NCOLBV,     &
        NEQNS,  NEQN1,  MTFU,   MTFL,    MBLOCK, MP,     MDF,        &
        KBOUN,  KKK,    IOUT,   ITP12,   ITP13,  NBLOCK, IGS,        &
        MDF1,   ISTAG,  LCOM,   NODLAY, NEQLAY, KLAY,   LEMBNK)
!*********************************************************************
!     TO SOLVE SYMMETRIC AND UNSYMMETRIC FINITE ELEMENT STATIC
!     EQUILIBRIUM EQUATIONS OUT-OF-CORE,
!     USING COMPACTED STORAGE AND COLUMN REDUCTION SCHEME
!     TREATING WITH BOUNDARY CONDITIONS.
!
!     IF   KKK=1   FOR TRIANGULARIZATION .
!     IF   KKK=2   FOR REDUCTION OF LOAD VECTOR.
!     IF   KKK=3   FOR BACK-SUSTITUTION.
!
!     31 JANUARY 2005
!     DEVELOPED BY  JAE-SUN  KANG
!     DEPARTMENT OF CIVIL ENGINEERING,     ENGINEERING COLLEGE
!     KYUNG  NAM  UNIVERSITY
!     MASAN,  KOREA (631-701)
!*********************************************************************
      USE KCHECK_COM
      USE NSPAC_COM
      USE NSPAC1_COM
      USE ICONT_COM
      USE ICONT1_COM
      USE NSSOL_COM
      USE VAR_COM
      USE CTIMS_COM
      USE CTMAS_COM
!*********************************************************************
      IMPLICIT DOUBLE PRECISION (A - H, O - Z)
!*********************************************************************
      DIMENSION  &
       V (NEQNS), STIFU (MTFU), STIFFU (MTFU), STIFL (MTFL), &
       STIFFL (MTFL), DIAGN (NEQNS), FIXED (NEQNS), FIXEB (NEQNS), &
       FIXEC (NEQNS), MAXAJ (NEQN1), NCOLBV (MBLOCK), IFFIX (NEQNS)
      DIMENSION GASHT (IGS, MDF1, MP), DIS (MDF, MP)
      DIMENSION NODLAY(MP), NEQLAY(NEQNS), ISTAG (NEQNS)
!*********************************************************************
!     CHECK OF ARRAY
!*********************************************************************
      WRITE (KCHK1, 7000)
 7000 FORMAT(/,1X,'*** AT UNSOL     ***')
!     WRITE (KCHK1, 7001)   &
!      NEQNS, MTFU, MTFL, NEQNS, NEQN1, MBLOCK, IGS, &
!      MDF, MP
 7001 FORMAT( &
       1X,'NEQNS  =',I5,1X,'MTFU    =',I5,1X,'MTFL    =',I5,/, &
       1X,'NEQNS  =',I5,1X,'MBLOCK =',I5,1X,'IGS     =',I5,/, &
       1X,'MDF    =',I5,1X,'MP      =',I5)
!*********************************************************************
      PRINT  *, 'KLAY =', KLAY
      WRITE (IOUT, 1199)
```

```
      WRITE (IOUT, 2299)  (NEQLAY(IEQNS), IEQNS=1,NEQNS)
 1199 FORMAT(//,2X,'LAYER OF EQUATIONS',/)
 2299 FORMAT(9(1X,I4))
!---------------------------------------------------------------------
      PRINT * , 'NINSCH =', NINSCH
      PRINT * , 'ITER   =', ITER
      PRINT * , 'UNSOL START'
      PRINT * , 'KKK    =', KKK
      NSTIF = ITP12
      NRED  = ITP13
      KHBB  = 0
      NEQNC = NEQNS + 1
!---------------------------------------------------------------------
      IF (KKK == 1) GO TO 10
      IF (KKK == 2) GO TO 610
      IF (KKK == 3) GO TO 750
!---------------------------------------------------------------------
   10 CONTINUE
!---------------------------------------------------------------------
!     CALCULATE DIAGONAL ELEMENTS OF STIFFNESS MATRIX OF DIAGN
!---------------------------------------------------------------------
      DO  NB = 1, NBLOCK  !  5000
!*********************************************************************
        IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM == 1)  &
          READ (NSTIF, REC = NB) (STIFU (JA), JA = 1, MTFU)
        IF (ISYMM == 1) CALL STCND (TEND, LCOM)
        IF (ISYMM == 1)  &
          ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!*********************************************************************
!*********************************************************************
        IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM /= 1)  &
          READ (NSTIF, REC = NB) (STIFU (JA), JA = 1, MTFU),    &
                                 (STIFL (JA), JA = 1, MTFL)
        IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
        IF (ISYMM /= 1)  &
          ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!*********************************************************************
        NCOLB = NCOLBV (NB)
        MM = MAXAJ (KHBB + 1) - 1
        DO  NC = 1, NCOLB  !  5100
!*********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*********************************************************************
          IF (NEQLAY(KHBB+ NC) > KLAY)  GO TO 5100
!*********************************************************************
          KN = MAXAJ (KHBB + NC) - MM
          DIAGN (KHBB + NC) = STIFU (KN)
 5100   CONTINUE
        END DO  !  5100
        KHBB = KHBB + NCOLB
      END DO  !  5000
!*********************************************************************
!     CHECK DIAGONAL TERM OF GLOBAL STIFFNESS MATRIX
!*********************************************************************
      WRITE (KCHK1, 8000)
      DO IW = 1, NEQNS
!*********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*********************************************************************
```

```
        IF (NEQLAY(IW) <= KLAY)  THEN
        IF (IFFIX (IW) >= 0) WRITE (KCHK1, 8010) IW, DIAGN (IW)
        END IF
      END DO
!-----------------------------------------------------------------------
!     FORMATS
!-----------------------------------------------------------------------
 8000 FORMAT(//,  &
       2X,'DIAGONAL ELEMENTS OF GLOBAL STIFFNESS MATRIX',/)
 8010 FORMAT(1X,I7,1X,E17.7)
!-----------------------------------------------------------------------
!     FACTORIZE STIFFNESS MATRIX  (LOOP OVER ALL BLOCKS)
!-----------------------------------------------------------------------
      KHBB = 0
      DO  NJ = 1, NBLOCK  !  600
!*********************************************************************
        IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM == 1)  &
          READ (NSTIF, REC = NJ) (STIFU (JA), JA = 1, MTFU)
        IF (ISYMM == 1) CALL STCND (TEND, LCOM)
        IF (ISYMM == 1)  &
          ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!*********************************************************************
        IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM /= 1)  &
          READ (NSTIF, REC = NJ) (STIFU (JA), JA = 1, MTFU),    &
                                 (STIFL (JA), JA = 1, MTFL)
        IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
        IF (ISYMM /= 1)  &
          ST12S (NSTEP) = ST12S (NSTEP) + (TEND-TSTART)
!*********************************************************************
        NCOLB = NCOLBV (NJ)
        MM = MAXAJ (KHBB +  1) - 1
!-----------------------------------------------------------------------
        IF (NJ == 1) GO TO 300
!-----------------------------------------------------------------------
        IM = 0
        IK = 1
        NJ1 = NJ - 1
!-----------------------------------------------------------------------
!     REDUCE BLOCK BY THE PRECEDING COUPLING BLOCKS
!-----------------------------------------------------------------------
        DO  NK = IK, NJ1  !  160
!-----------------------------------------------------------------------
!*****  DIRECT ACCESS  *****
!*********************************************************************
          IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
          IF (ISYMM == 1)  &
            READ (NRED, REC = NK) (STIFFU (JB), JB = 1, MTFU)
          IF (ISYMM == 1) CALL STCND (TEND, LCOM)
          IF (ISYMM == 1)  &
            ST13S (NSTEP) = ST13S (NSTEP) + (TEND- TSTART)
!*********************************************************************
          IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
          IF (ISYMM /= 1)    &
            READ (NRED, REC = NK) (STIFFU (JB), JB = 1, MTFU), &
                                  (STIFFL (JB), JB = 1, MTFL)
          IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
          IF (ISYMM /= 1)  &
            ST13S (NSTEP) = ST13S (NSTEP) + (TEND- TSTART)
!*********************************************************************
!*****  DIRECT ACCESS  *****
!-----------------------------------------------------------------------
```

```
          IMM = 0
          DO   KA = 1, NK   !   100
            IMM = IMM +  NCOLBV (KA)
          END DO   !   100
          KHB = KHBB - IMM
!-----------------------------------------------------------------------
          MC = MAXAJ (IM +  1) - 1
!-----------------------------------------------------------------------
          DO   N = 1, NCOLB   !   200
            IF (NSTAG == 1.AND.NTHERM == 1) THEN
              IF (ITER /= NITER.AND.ISTAG (KHBB +  N) == 2) GO TO 200
              IF (ITER == NITER.AND.ISTAG (KHBB +  N) == 1) GO TO 200
            END IF
!**********************************************************************
!      2005-03-25 FRIDAY
!      CONSTRUCTION LAYER
!**********************************************************************
            IF (NEQLAY(KHBB+ N) > KLAY)   GO TO 200
!**********************************************************************
            IF (KBOUN == 0) GO TO 3100
            IF (IFFIX (KHBB +  N) < 0) GO TO 200
 3100        CONTINUE
            KN = MAXAJ (KHBB +  N) - MM
            KL = KN +  1
            KU = MAXAJ (KHBB +  N +  1) - 1 - MM
            KH = KU - KL - N +  1
            KC = KH - KHB
            KS = N +  KHBB
            IF (KS <= NEQNC) GO TO 205
            KDIF = KU - KL
            KK = KS - NEQNC - 1
            IF (KDIF < KK) GO TO 200
  205       IF (KC <= 0) GO TO 200
            IC = 0
            KCL = NCOLBV (NK) - KC +  1
            IF (KCL > 0) GO TO 210
            IC = 1 - KCL
            KCL = 1
  210       KCR = NCOLBV (NK)
            KLT = KU - IC
            DO   K = KCL, KCR   !   220
              IC = IC +  1
              KLT = KLT - 1
!**********************************************************************
              IF (NSTAG == 1.AND.NTHERM == 1) THEN
                IF (ITER /= NITER.AND.ISTAG (K +  IM) == 2) GO TO 220
                IF (ITER == NITER.AND.ISTAG (K +  IM) == 1) GO TO 220
              END IF
!**********************************************************************
!      2005-03-25 FRIDAY
!      CONSTRUCTION LAYER
!**********************************************************************
              IF (NEQLAY(K+ IM) > KLAY)   GO TO 220
!**********************************************************************
              IF (KBOUN == 0) GO TO 3102
              IF (IFFIX (K +  IM) < 0) GO TO 220
 3102          CONTINUE
              KI = MAXAJ (K +  IM) - MC
              ND = MAXAJ (K +  IM +  1) - KI - MC - 1
!-----------------------------------------------------------------------
!CCC   IF (ND)   220,220,230
              IF (ND > 0) GO TO 230
              IF (ND <= 0) GO TO 220
```

```
!-----------------------------------------------------------------------
  230          KK = MIN0 (IC, ND)
               C = 0.D0
               IF (ISYMM /= 1) D = 0.D0
               JJ = 1
               IF (K + IM <= NEQNC) GO TO 235
               JJ = K + IM - NEQNC
               IF (KK < JJ) GO TO 220
  235          DO  L = JJ, KK  !  240
                 IF (NSTAG == 1.AND.NTHERM == 1) THEN
                   IF (ITER /= NITER.AND.ISTAG (K + IM - L) == 2)  &
                     GO TO 240
                   IF (ITER == NITER.AND.ISTAG (K + IM - L) == 1)  &
                     GO TO 240
                 END IF
!***********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!***********************************************************************
                 IF (NEQLAY(K+ IM-L) > KLAY)  GO TO 240
!***********************************************************************
                 IF (KBOUN == 0) GO TO 3104
                 IF (IFFIX (K + IM - L) < 0) GO TO 240
 3104             CONTINUE
                 IF (ISYMM == 1)    &
                   C = C + STIFFU (KI + L) * STIFU (KLT + L)
                 IF (ISYMM /= 1)    &
                   C = C + STIFFL (KI + L) * STIFU (KLT + L)
                 IF (ISYMM /= 1)    &
                   D = D+ STIFFU (KI + L) * STIFL (KLT + L)
  240          CONTINUE
               END DO  !  240
               STIFU (KLT) = STIFU (KLT) - C
               IF (ISYMM /= 1) STIFL (KLT) = STIFL (KLT) - D
  220        CONTINUE
             END DO  !  220
  200      CONTINUE
           END DO  ! 200
!-----------------------------------------------------------------------
       IM = IM + NCOLBV (NK)
!-----------------------------------------------------------------------
       END DO  !  160
!-----------------------------------------------------------------------
!     REDUCE BLOCK BY ITSELF
!-----------------------------------------------------------------------
  300  DO  N = 1, NCOLB  !  400
!***********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!***********************************************************************
         IF (NEQLAY(KHBB+ N) > KLAY)  GO TO 400
!***********************************************************************
         IF (NSTAG == 1.AND.NTHERM == 1) THEN
           IF (ITER /= NITER.AND.ISTAG (KHBB + N) == 2) GO TO 400
           IF (ITER == NITER.AND.ISTAG (KHBB + N) == 1) GO TO 400
         END IF
!***********************************************************************
         IF (KBOUN == 0) GO TO 3106
         IF (IFFIX (KHBB + N) < 0) GO TO 400
 3106     CONTINUE
         KN = MAXAJ (KHBB + N) - MM
         KL = KN + 1
         KU = MAXAJ (KHBB + N + 1) - 1 - MM
```

```
          KDIF = KU - KL
          KH = MIN0 (KDIF, N - 1)
          KS = N + KHBB
          IF (KDIF < KS - NEQNC - 1) GO TO 400
!-----------------------------------------------------------------------
!CCC   IF (KH)  550,440,460
          IF (KH > 0) GO TO 460
          IF (KH == 0) GO TO 440
          IF (KH < 0) GO TO 550
!-----------------------------------------------------------------------
  460      K = N - KH
          KLT = KL + KH
          IC = 0
          IF ( (N - 1) < KDIF) IC = KDIF - N + 1
!-----------------------------------------------------------------------
          DO  J = 1, KH  !  480
            IC = IC + 1
            KLT = KLT - 1
!**********************************************************************
            IF (NSTAG == 1.AND.NTHERM == 1) THEN
              IF (ITER /= NITER.AND.ISTAG (KHBB + K) == 2) GO TO 480
              IF (ITER == NITER.AND.ISTAG (KHBB + K) == 1) GO TO 480
            END IF
!**********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!**********************************************************************
            IF (NEQLAY(KHBB+ K) > KLAY)  GO TO 480
!**********************************************************************
            IF (KBOUN == 0) GO TO 3108
            IF (IFFIX (KHBB + K) < 0) GO TO 480
 3108        CONTINUE
            KI = MAXAJ (KHBB + K) - MM
            ND = MAXAJ (KHBB + K + 1) - KI - MM - 1
!-----------------------------------------------------------------------
!CCC   IF (ND)  480,480,500
            IF (ND > 0) GO TO 500
            IF (ND <= 0) GO TO 480
!-----------------------------------------------------------------------
 500        KK = MIN0 (IC, ND)
            C = 0.D0
            IF (ISYMM /= 1) D = 0.D0
            JJ = 1
            IF (K + KHBB <= NEQNC) GO TO 510
            JJ = K + KHBB - NEQNC
            IF (KK < JJ) GO TO 480
 510        DO  L = JJ, KK  !  520
!**********************************************************************
              IF (NSTAG == 1.AND.NTHERM == 1) THEN
                IF (ITER /= NITER.AND.ISTAG (KHBB + K - L) == 2)  &
                  GO TO 520
                IF (ITER == NITER.AND.ISTAG (KHBB + K - L) == 1)  &
                  GO TO 520
              END IF
!**********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!**********************************************************************
              IF (NEQLAY(KHBB+ K-L) > KLAY)  GO TO 520
!**********************************************************************
              IF (KBOUN == 0) GO TO 3110
              IF (IFFIX (KHBB + K - L) < 0) GO TO 520
 3110          CONTINUE
```

```
            IF (ISYMM == 1) C = C +  STIFU (KI +  L) * STIFU (KLT +  L)
            IF (ISYMM /= 1) C = C +  STIFL (KI +  L) * STIFU (KLT +  L)
            IF (ISYMM /= 1) D = D+ STIFU (KI +  L) * STIFL (KLT +  L)
  520      END DO   !  520
           STIFU (KLT) = STIFU (KLT) - C
           IF (ISYMM /= 1) STIFL (KLT) = STIFL (KLT) - D
  480      CONTINUE
           K = K +  1
         END DO   !  480
!-----------------------------------------------------------------------
  440    K = KS
         IF (KS <= NEQNC) GO TO 450
         K = NEQNC +  1
         JJ = KS - NEQNC - 1
         KL = KL + JJ
         KH = KU - KL
!-----------------------------------------------------------------------
!CCC   IF (KH)  400,450,450
         IF (KH >= 0) GO TO 450
         IF (KH < 0) GO TO 400
!-----------------------------------------------------------------------
  450    E = 0.D0
         DO  KK = KL, KU   !  540
           K = K - 1
!*********************************************************************
           IF (NSTAG == 1.AND.NTHERM == 1) THEN
             IF (ITER /= NITER.AND.ISTAG (K) == 2) GO TO 540
             IF (ITER == NITER.AND.ISTAG (K) == 1) GO TO 540
           END IF
!*********************************************************************
!      2005-03-25 FRIDAY
!      CONSTRUCTION LAYER
!*********************************************************************
           IF (NEQLAY(K) > KLAY)  GO TO 540
!*********************************************************************
           IF (KBOUN == 0) GO TO 3112
           IF (IFFIX (K) < 0) GO TO 540
 3112       CONTINUE
           C = STIFU (KK) / DIAGN (K)
           IF (ISYMM /= 1) D = STIFL (KK) / DIAGN (K)
           IF (ISYMM == 1) E = E+ C * STIFU (KK)
           IF (ISYMM /= 1) E = E+ C * STIFL (KK)
           STIFU (KK) = C
           IF (ISYMM /= 1) STIFL (KK) = D
  540     CONTINUE
         END DO   !  540
         STIFU (KN) = STIFU (KN) - E
         IF (ISYMM /= 1) STIFL (KN) = STIFL (KN) - E
         DIAGN (KS) = STIFU (KN)
!-----------------------------------------------------------------------
  420    CONTINUE
!CCC   IF (KS - NEQNC)  550,550,400
         IF ( (KS - NEQNC) <= 0) GO TO 550
         IF ( (KS - NEQNC) > 0) GO TO 400
!-----------------------------------------------------------------------
  550    DIAGN (KS) = STIFU (KN)
!-----------------------------------------------------------------------
!CCC   IF (DIAGN(KS))  400,555,400
         IF (DIAGN (KS) == 0.0) GO TO 555
         IF (DIAGN (KS) /= 0.0) GO TO 400
!-----------------------------------------------------------------------
  555    CONTINUE
  560    WRITE (IOUT, 2000) KS, DIAGN (KS)
```

```
          STOP
!-----------------------------------------------------------------------
  400     CONTINUE
        END DO  !  400
!-----------------------------------------------------------------------
      KHBB = KHBB + NCOLB
!*********************************************************************
!*****  DIRECT ACCESS  *****
!*********************************************************************
      IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
      IF (ISYMM == 1)   &
        WRITE (NRED, REC = NJ) (STIFU (JA), JA = 1, MTFU)
      IF (ISYMM == 1) CALL STCND (TEND, LCOM)
      IF (ISYMM == 1)  &
        ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*********************************************************************
      IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
      IF (ISYMM /= 1)  &
        WRITE (NRED, REC = NJ) (STIFU (JA), JA = 1, MTFU),    &
                               (STIFL (JA), JA = 1, MTFL)
      IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
      IF (ISYMM /= 1)  &
        ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*********************************************************************
!*****  DIRECT ACCESS  *****
!*********************************************************************
      END DO  !  600
!-----------------------------------------------------------------------
      PRINT * , 'UNSOL END'
      PRINT * , 'KKK =', KKK
      RETURN
!-----------------------------------------------------------------------
!     SOLUTION OF EQUATIONS (LOOP OVER ALL BLOCKS)
!-----------------------------------------------------------------------
!     REDUCE THE LOAD VECTOR
!-----------------------------------------------------------------------
  610 KHBB = 0
      NRED = ITP13
      DO  NJ = 1, NBLOCK  !  700
!*****  DIRECT ACCESS  *****
!*********************************************************************
        IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM == 1)  &
          READ (NRED, REC = NJ) (STIFU (JA), JA = 1, MTFU)
        IF (ISYMM == 1) CALL STCND (TEND, LCOM)
        IF (ISYMM == 1)  &
          ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*********************************************************************
!*********************************************************************
        IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM /= 1)  &
          READ (NRED, REC = NJ) (STIFU (JB), JB = 1, MTFU),    &
                                (STIFL (JB), JB = 1, MTFL)
        IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
        IF (ISYMM /= 1)  &
          ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*********************************************************************
!*****  DIRECT ACCESS  *****
  710   NCOLB = NCOLBV (NJ)
        MM = MAXAJ (KHBB + 1) - 1
        DO  N = 1, NCOLB  !  720
          IF (NSTAG == 1.AND.NTHERM == 1) THEN
            IF (ITER /= NITER.AND.ISTAG (KHBB + N) == 2) GO TO 720
```

```
          IF (ITER == NITER.AND.ISTAG (KHBB +  N) == 1) GO TO 720
        END IF
!*****************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*****************************************************************
        IF (NEQLAY(KHBB+ N) > KLAY)   GO TO 720
!*****************************************************************
        IF (KBOUN == 0) GO TO 4103
        IF (IFFIX (KHBB +  N) < 0) GO TO 720
 4103    CONTINUE
        KL = MAXAJ (N +  KHBB) - MM +  1
        KU = MAXAJ (N +  KHBB +  1) - MM - 1
        KS = N +  KHBB
        K = KS
        IF (KS <= NEQNC) GO TO 725
        K = NEQNC +  1
        KL = KL +  KS - NEQNC - 1
  725    CONTINUE
!-----------------------------------------------------------------
!CCC   IF (KU - KL)   720,730,730
        IF ( (KU - KL) >= 0) GO TO 730
        IF ( (KU - KL) < 0) GO TO 720
!-----------------------------------------------------------------
  730    C = 0.D0
        DO   KK = KL, KU   !   740
          K = K - 1
          IF (NSTAG == 1.AND.NTHERM == 1) THEN
            IF (ITER /= NITER.AND.ISTAG (K) == 2) GO TO 740
            IF (ITER == NITER.AND.ISTAG (K) == 1) GO TO 740
          END IF
!*****************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*****************************************************************
          IF (NEQLAY(K) > KLAY)    GO TO 740
!*****************************************************************
          IF (KBOUN == 0) GO TO 3116
          IF (IFFIX (K) < 0) GO TO 740
 3116      CONTINUE
          IF (ISYMM == 1) C = C +  STIFU (KK) * V (K)
          IF (ISYMM /= 1) C = C +  STIFL (KK) * V (K)
  740      CONTINUE
!-----------------------------------------------------------------
        END DO
!-----------------------------------------------------------------
        V (KS) = V (KS) - C
  720   CONTINUE
!-----------------------------------------------------------------
      END DO
!-----------------------------------------------------------------
      KHBB = KHBB +  NCOLB
!-----------------------------------------------------------------
    END DO
!-----------------------------------------------------------------
    PRINT * , 'UNSOL END'
    PRINT * , 'KKK =', KKK
    RETURN
!-----------------------------------------------------------------
!     BACKSUBSTITUTE
!-----------------------------------------------------------------
  750 CONTINUE
    DO   N = 1, NEQNS   !   7900
```

```
        IF (NSTAG == 1.AND.NTHERM == 1) THEN
          IF (ITER /= NITER.AND.ISTAG (N) == 2) GO TO 7900
          IF (ITER == NITER.AND.ISTAG (N) == 1) GO TO 7900
        END IF
!*****************************************************************
!       2005-03-25 FRIDAY
!       CONSTRUCTION LAYER
!*****************************************************************
        IF (NEQLAY(N) > KLAY)  GO TO 7900
!*****************************************************************
        IF (KBOUN == 0) GO TO 4307
        IF (IFFIX (N) < 0) GO TO 7900
 4307   CONTINUE
        V (N) = V (N) / DIAGN (N)
 7900 CONTINUE
      END DO  !  7900
!-----------------------------------------------------------------------
      KHBB = NEQNS
      NBL = NBLOCK
      DO  NJ = 1, NBLOCK  !  800
        NCOLB = NCOLBV (NBL)
        IF (NBLOCK == 1) GO TO 820
        NJB1 = NBLOCK - NJ + 1
!*****  DIRECT ACCESS  *****
!*****************************************************************
        IF (ISYMM == 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM == 1)  &
          READ (NRED, REC = NJB1) (STIFU (JA), JA = 1, MTFU)
        IF (ISYMM == 1) CALL STCND (TEND, LCOM)
        IF (ISYMM == 1)  &
          ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*****************************************************************
        IF (ISYMM /= 1) CALL STCND (TSTART, LCOM)
        IF (ISYMM /= 1)  &
          READ (NRED, REC = NJB1) (STIFU (JB), JB = 1, MTFU),    &
                                  (STIFL (JB), JB = 1, MTFL)
        IF (ISYMM /= 1) CALL STCND (TEND, LCOM)
        IF (ISYMM /= 1)  &
          ST13S (NSTEP) = ST13S (NSTEP) + (TEND-TSTART)
!*****************************************************************
!       DIRECT ACCESS
!-----------------------------------------------------------------------
  820   KHBB = KHBB - NCOLB
        MM = MAXAJ (KHBB + 1) - 1
        N = NCOLB
        DO  L = 1, NCOLB  !  860
          NL1 = N - L + 1
          IF (NSTAG == 1.AND.NTHERM == 1) THEN
            IF (ITER /= NITER.AND.ISTAG (KHBB + NL1) == 2)  &
              GO TO 860
            IF (ITER == NITER.AND.ISTAG (KHBB + NL1) == 1)  &
              GO TO 860
          END IF
!*****************************************************************
!       2005-03-25 FRIDAY
!       CONSTRUCTION LAYER
!*****************************************************************
          IF (NEQLAY(KHBB+ NL1) > KLAY)  GO TO 860
!*****************************************************************
          IF (KBOUN == 0) GO TO 3213
          IF (IFFIX (KHBB + NL1) < 0) GO TO 860
 3213     CONTINUE
          KL = MAXAJ (KHBB + NL1) - MM + 1
```

```
          KU = MAXAJ (KHBB + NL1 + 1) - MM - 1
          KS = KHBB + NL1
          IF (KS <= NEQNC) GO TO 850
          K = NEQNC + 1
          KL = KL + KS - NEQNC - 1
!-----------------------------------------------------------------------
  850     CONTINUE
!CCC   IF (KU-KL)  860,890,890
          IF ( (KU - KL) >= 0) GO TO 890
          IF ( (KU - KL) < 0) GO TO 860
!-----------------------------------------------------------------------
  890     K = KS
          DO  KK = KL, KU  !  900
            K = K - 1
            IF (NSTAG == 1.AND.NTHERM == 1) THEN
              IF (ITER /= NITER.AND.ISTAG (K) == 2) GO TO 900
              IF (ITER == NITER.AND.ISTAG (K) == 1) GO TO 900
            END IF
!*********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*********************************************************************
            IF (NEQLAY(K) > KLAY)  GO TO 900
!*********************************************************************
            IF (KBOUN == 0) GO TO 3215
            IF (IFFIX (K) < 0) GO TO 900
 3215        CONTINUE
            V (K) = V (K) - STIFU (KK) * V (KS)
  900      CONTINUE
          END DO  !  900
  860    CONTINUE
        END DO  !  860
        NBL = NBL - 1
      END DO  !  800
!-----------------------------------------------------------------------
!     CHECK DISPLACEMENTS AND BOUNDARY VALUES
!     19:02 TUE JULY 08 (2003)
!-----------------------------------------------------------------------
      WRITE (IOUT, 9105)
      DO  IEQNS = 1, NEQNS  !  1301
        IF (NSTAG == 1.AND.NTHERM == 1) THEN
          IF (ITER /= NITER.AND.ISTAG (IEQNS) == 2) GO TO 1301
          IF (ITER == NITER.AND.ISTAG (IEQNS) == 1) GO TO 1301
        END IF
!*********************************************************************
!     2005-03-25 FRIDAY
!     CONSTRUCTION LAYER
!*********************************************************************
        IF (NEQLAY(IEQNS) > KLAY)  GO TO 1302
!*********************************************************************
        IF (IFFIX (IEQNS) < 0) V (IEQNS) = FIXED (IEQNS)
        WRITE (IOUT, 9107)    &
             IEQNS, IFFIX (IEQNS), V (IEQNS), FIXED (IEQNS)
        GO TO 1301
!*********************************************************************
!     CLEAR TO ZERO FOR NOT CONSTRUCTED LAYER EQUATIONS
!*********************************************************************
 1302   CONTINUE
        V(IEQNS) = 0.D0
!*********************************************************************
 1301 CONTINUE
      END DO  !  1301
!*********************************************************************
```

```
 9105 FORMAT(//, &
       2X,'== DISPLACEMENTS AND BOUNDARY CONDITIONS ==',/, &
       7X,'NU.EQN.',2X,'IFFIX',2X,'DISPLACEMENT',   &
       2X,'BOUNDARY VALUE', /)
 9107 FORMAT( &
       2X,I9,2X,I9,2X,E15.7,2X,E15.7)
!*********************************************************************
!      2005-04-28
!      CLEAR PORE PRESSURE TO ZERO FOR ENBANKMENTS
!*********************************************************************
       DO IEQNS = 1, NEQNS
!!!     IF (NEQLAY (IEQNS) > KLAY)   GO TO 9987
         IF (NEQLAY (IEQNS) <= LEMBNK)   GO TO 9987
           NPOR = MOD (IEQNS, NDF)
           IF (NPOR == 0)   V (IEQNS) = 0.D0
           IF (NPOR == 0)      &
             PRINT *, 'IEQNS =',IEQNS, 'V(IEQNS)=',V(IEQNS),    &
                      'NPOR   =',NPOR
 9987 CONTINUE
       END DO
!*********************************************************************
!      CLEAR DISPLACEMENTS MATRIX DIS(NP,NDF) TO ZERO
!      AT FIRST ITERATION (8TH MAY 2003, 22:22)
!*********************************************************************
       IF (ITER == 1) THEN
         DO II = 1, NP
           DO JJ = 1, NDF
             DIS (JJ, II) = 0.D0
           END DO
         END DO
       END IF
!---------------------------------------------------------------------
!      TRANSFORM V-DISPLACEMENT TO DIS-ARRAY
!---------------------------------------------------------------------
       IEQ = 0
       DO   II = 1, NP   !   1101
         DO   JJ = 1, NDF   !   1101
           IEQ = IEQ + 1
!*********************************************************************
           IF (NSTAG == 1.AND.NTHERM == 1) THEN
             IF (ITER /= NITER.AND.ISTAG (IEQ) == 2) CYCLE
             IF (ITER == NITER.AND.ISTAG (IEQ) == 1) CYCLE
           END IF
!*********************************************************************
           DIS (JJ, II) = V (IEQ)
         END DO   !   1101
       END DO   !   1101
!*********************************************************************
!      CHECK OF SOLUTION BY PRFILE STAGGED SOLVER
!*********************************************************************
       IF (NTHERM == 0)   WRITE (IOUT,   3101) NSTEP, ITER
       IF (NTHERM == 1)   WRITE (IOUT,   4102) NSTEP, ITER
       DO   II = 1, NP   !   1301
         IF (NDF == 3)    &
           WRITE (IOUT,   3210) II, (DIS (JJ, II), JJ = 1, NDF),    &
                                NODLAY(II)
         IF (NDF == 4)    &
           WRITE (IOUT,   3211) II, (DIS (JJ, II), JJ = 1, NDF),    &
                                NODLAY(II)
       END DO   !   1301
!*********************************************************************
!      TRANSFORM DIS(NODAL DISPLACEMENT) TO GASHT(3,NDF1,NP) FOR
!      PRINTING
```

```
!-----------------------------------------------------------------------
      IF (NINSCH < 1) GO TO 2004
      IF (ITER /= NITER) GO TO 2004
      DO   N = 1, NP   !  2003
        GASHT (3, NDF, N) = DIS (NDF, N)
      END DO   !  2003
      GO TO 2005
 2004 NDF1 = NDF - 1
      IF (NTHERM == 1) NDF1 = NDF1 - 1
      DO   N = 1, NP   !  661
        DO   I = 1, NDF1   !  660
          IF (NINSCH < 1) GO TO 659
          GASHT (3, I, N) = DIS (I, N)
          GO TO 660
  659      GASHT (2, I, N) = DIS (I, N)
  660      CONTINUE
        END DO   !  660
      IF (NINSCH < 1) GO TO 673
      IF (NTHERM == 1)   &
        GASHT (3, NDF - 1, N) = DIS (NDF - 1, N) * CURILL
      GO TO 661
  673 IF (NTHERM == 1)    &
        GASHT (2, NDF - 1, N) = DIS (NDF - 1, N)   * CURILL
      GASHT (2, NDF, N) = DIS (NDF, N) * CU2
      IF (NTHERM == 0) GASHT (2, NDF, N) = DIS (NDF, N) * CURILL
  661 CONTINUE
      END DO   !  661
 2005 CONTINUE
!-----------------------------------------------------------------------
!     FORMATS
!-----------------------------------------------------------------------
 3101 FORMAT &
       (//,1X,'SOLUTION OF EQUATION EVERY ITERRATION',2I5,/, &
        2X,'POINT',2X,'X-DISPLACEMENT',2X,'Y-DISPLACEMENT', &
        2X,'PORE PRESSURE',2X,'LAYER',/)
 4102 FORMAT &
       (//,1X,'SOLUTION OF EQUATION EVERY ITERRATION',2I5,/, &
         2X,'POINT',2X,'X-DISPLACEMENT',2X,'Y-DISPLACEMENT', &
         2X,'PORE PRESSURE',2X,' TEMPERATURE ',2X,'LAYER',/)
 3210 FORMAT &
       (2X,I5,3(1X,D14.7),1X,I4)
 3211 FORMAT &
       (2X,I9,4(1X,D14.7),1X,I4)
 2000 FORMAT &
       (//,2X,'STOP - STIFFNESS NOT DEFINITE',//,2X, &
         'ZERO - PIVOT FOR EQUATION = ',I9,//,2X, &
         'PIVOT =   ',D20.12)
!-----------------------------------------------------------------------
      PRINT * , 'UNSOL END'
      PRINT * , 'KKK =', KKK
      RETURN
      END SUBROUTINE UNSOL
```